	Logical Record Architecture for Health and Social Care: Data Types Specification			
	Programme	NPFIT	Document Record ID Key	
	Sub-Prog / Project	Logical record architecture	NPFIT-FNT-TO-DPM-0936.03	
	Prog. Director	David Perry	Status	Draft for comment
	Owner	S. Bentley	Version	0.3.0
	Author	Rob Challen J. Arnett	Version Date	2009-07-29

Logical Record Architecture for Health and Social Care: Data Types Specification

Amendment History:

Version	Date	Amendment History
0.1		This document replaces the Logical Record Architecture for Health and Social Care: Abstract Data Types Specification, on which it is loosely based. It is a new document and hence starting at new version.
0.2	2009-04-29	Changes in response to internal comments. Including reordering document to align to original specification, adjustments to naming conventions, removal of some classes included from old version of underlying specification. Clarification of use of CD data type attributes. Addition of appendix to elaborate LRA allowed representational forms of codes.
0.3.0	2009-07-29	Section 5.1.4: added Class CD.CV.SCT (SNOMED CT Coded data – no translations)
		Section 5.1.2 para 2: removed text "As a design principle, although appealing, it was decided not to specialise CD class for different code system subtypes, to improve conformance to ISO 21090. The down side of this is some necessary overloading of the attributes."
		Removed Appendix B – Enumerations as information now specified as part of LRA content release.
		Removed Appendix C – ISO Dates and Time Representation as the LRA conforms ISO 21090 date and time representation.

Forecast Changes:

Anticipated Change	When

Reviewers:

This document must be reviewed by the following:<author to indicate reviewers>

Name	Signature	Title / Responsibility	Date	Version

Approvals:

This document must be approved by the following: <author to indicate approvers>

Name	Signature	Title / Responsibility	Date	Version

Distribution:

Available to all LRA Stakeholders

Document Status:

This is a controlled document.

Whilst this document may be printed, the electronic version maintained in FileCM is the controlled copy. Any printed copies of the document are not controlled.

Related Documents:

These documents will provide additional information.

Ref no	Doc Reference Number	Title	Version
1	NPFIT-SHR-QMS-PRP-0015	Glossary of Terms Consolidated.doc	<enter latest>

Glossary of Terms:

List any new terms created in this document. Mail the NPO Quality Manager to have these included in the master glossary above [1].

Term	Acronym	Definition

Contents

1	Background	7
1.1	Purpose	7
1.2	Audience	7
1.3	Scope	7
1.4	Format	8
2	Introduction	8
2.1	Data Type Definitions	8
2.2	Semantic Scope	8
2.3	ISO Notation	9
2.4	Data type abbreviations	9
3	Basic Data Types	11
3.1	Package Ira.datatypes.ISO 21090	11
3.1.1	Class HXIT (history)	11
3.1.2	Class ANY (any data)	12
3.1.3	Class BL (Boolean)	13
3.1.4	Class BL.NonNull (Boolean – non null)	13
4	Text and Binary Data Types	14
4.1	Package Ira.datatypes.ISO 21090	14
4.1.1	Class ED (Encapsulated data)	14
4.1.2	Class ED.Doc (Encapsulated data – documentation)	17
4.1.3	Class ED.Doc.Inline (Encapsulated data – documentation inline)	18
4.1.4	Class ED.Doc.Ref (Encapsulated data – documentation by reference)	18
4.1.5	Class ED.Image (Encapsulated data – image)	18
4.1.6	Class ED.Signature (Encapsulated data – electronic signature)	19
4.1.7	Class ED.Text (Encapsulated data – text)	19
4.1.8	Class ST (String)	20
4.1.9	Class ST.NT (String – no translations)	20
4.1.10	Class SC (String with code)	21
4.1.11	Class SC.NT (Coded string no translation)	22
5	Coded Datatypes (Terminology)	22
5.1	Package Ira.datatypes.ISO 21090	23
5.1.1	Class CS (Coded simple item)	23
5.1.2	Class CD (Coded data)	24
5.1.3	Class CD.CV (Coded data – no translations)	27
5.1.4	Class CD.CV.SCT (SNOMED CT Coded data – no translations)	27
6	Identity and Location Datatypes	28
6.1	Package Ira.datatypes.ISO 21090	29
6.1.1	Class II (Instance identifier)	29
6.1.2	Class TEL (Telecommunications locator)	30
6.1.3	Class TEL.Email (Email address)	31

6.1.4	Class TEL.Person (Person telecommunication locator)	31
6.1.5	Class TEL.Phone (Telephone number)	32
6.1.6	Class TEL.Url (Universal resource locator)	32
7	Name and Address Datatypes.....	32
7.1	Package Ira.datatypes.ISO 21090.....	33
7.1.1	Class ADXP (Address part)	33
7.1.2	Class AD (Address)	34
7.1.3	Class ENXP (Entity name part)	35
7.1.4	Class EN (Entity name)	36
7.1.5	Class EN.ON (Organisation Name)	37
7.1.6	Class EN.PN (Person Name)	37
7.1.7	Class EN.TN (Trivial Name)	38
8	Package Ira.datatypes.ISO 21090.quantitytypes.....	39
8.1	Package Ira.datatypes.ISO 21090.....	39
8.1.1	Abstract Class QTY (quantity)	39
8.1.2	Class INT (Integer quantity)	41
8.1.3	Class INT.NonNeg (Integer quantity – not negative value)	41
8.1.4	Class INT.Pos (Integer quantity – positive value)	41
8.1.5	Class CO (Coded ordinal quantity)	42
8.1.6	Class REAL (Real quantity)	43
8.1.7	Class RTO (Ratio quantity)	43
8.1.8	Class MO (Monetary quantity)	44
8.1.9	Class PQV (Physical quantity value)	44
8.1.10	Class PQ (Physical quantity with UCUM units)	45
8.1.11	Class PQ.Time (Physical quantity – time related)	46
8.1.12	Class PQR (Physical quantity with coded units)	46
8.1.13		47
8.1.14	Class TS (Time stamp)	47
8.1.15	Class TS.Birth (Time stamp – birth date)	48
8.1.16	Class TS.Date (Time stamp – date)	48
8.1.17	Class TS.Date.Full (Time stamp – fully specified date)	48
8.1.18	Class TS.DateTime (Time stamp – date and time)	49
8.1.19	Class TS.DateTime.Full (Time stamp – fully specific date and time)	49
9	Continuous Set Data types.....	49
9.1	Package Ira.datatypes.ISO 21090.....	49
9.1.1	Abstract Class QSET (Continuous set)	49
9.1.2	Class IVL (interval)	50
9.1.3	Class IVL.High (Interval – closed high)	52
9.1.4	Class IVL.Low (Interval – closed low)	52
9.1.5	Class IVL.Width (Interval - width only)	52
9.1.6	Class QSC (Coded quantity set)	53
9.1.7	Class QSI (intersection of continuous sets)	53
9.1.8	Class QSD (Continuous set – difference)	54
9.1.9	Class QSP (Continuous set – periodic hull)	54
9.1.10	Class QSS (Set of continuous sets)	54
9.1.11	Class QSU (Union of continuous sets)	55
9.1.12	Class PIVL (Periodic interval)	55

9.1.13	Class EIVL (Event based interval)	56
9.1.14	Class GTS.BoundedPIVL (Bounded periodic time interval)	57
10	Uncertainty Datatypes	57
10.1	Package lra.datatypes.ISO 21090	57
10.1.1	Class URG (Uncertain range)	57
10.1.2	Class URG.High (uncertain range, bounded at high end)	58
10.1.3	Class URG.Low (uncertain range, bounded at low end)	58
11	References	59
A	Appendix A – Value Types	61
	Primitive and Derived Value Types.....	Error! Bookmark not defined.
	Primitive Value Types	61
	String Derived Value Types	61
B	Appendix B – Enumerations	Error! Bookmark not defined.
C	Appendix B – Representational forms of codes in the LRA	62
	Requirements	62
	Representation of SNOMED CT and dm+d codes and coded expressions in the LRA	63
	Pre-coordinated codes (attested)	63
	Post-coordinated codes (attested – close to user)	64
	Normal form	64
	Representation of other code set codes	65

1 Background

ISO 21090 is a draft international standard specification of data types for healthcare interoperability. It defines its scope as to:

- provide set of data type definitions for representing and exchanging basic concepts that are commonly encountered in healthcare environments in support of information exchange in the healthcare environment,
- specify a collection of healthcare related data types suitable for use in a number of health related information environments,
- declare the semantics of these data types using the terminology, notations and data types defined in ISO 11404 rev 2005,
- provide UML definitions of the same data types using the terminology, notation and types defined in Unified Modelling Language (UML) version 2.0,
- define an XML (Extensible Mark up Language) based representation of the data types suitable for use when exchanging information between information processing entities.

1.1 Purpose

This is a constrained implementation profile, with some associated guidance for usage, of ISO 21090 for the specific purposes of the LRA. It is not a full explanation of the ISO 21090 data types, and an understanding of the data types is assumed. The baseline version of the standard that this profile is based on can be found at https://svn.connectingforhealth.nhs.uk/svn/public/lra/TRUNK/prod_env/ref/ISO_21090_20080317.pdf.

1.2 Audience

The intended audience of this specification is any individual, group or organisation involved in the development or use of the *LRA*.

1.3 Scope

This LRA data types profile aims to define the expected level of conformance to ISO 21090 that is expected of system interfaces that are conformant to the LRA. This is therefore a constrained subset of ISO 21090, and the standard has not been extended in any way. All the constraints of the classes in the standard are expressed in object constraint language. Some restrictions described here relate to patterns of use, or to specific vocabularies that the LRA will adopt. These constraints are described here but not formally expressed.

The LRA data types profile adopts implicitly the characterising operations specified by ISO 21090. The operations themselves, however, are not detailed in this specification and the reader should consult the international standard for further information.

An LRA conformant data service interface is not required to provide the operations (or access to the operations) but the software application (or system of applications) that provides the interface must interpret data passing through that interface in the context of the specifications for methods in ISO 21090. Data intended to represent

values of ISO 21090 data types carry with them, by virtue of this intent, processing obligations that are not evident in the data representation. An information processing entity which conforms directly to the LRA profile of ISO 21090 shall, to the extent that the entity provides operations other than movement or translation of values, define operations on the healthcare data types which can be derived from, or are otherwise consistent with the characterizing operations specified by ISO 21090.

1.4 Format

This specification consists of

- Introduction
- LRA ISO 21090 Data Types Implementation Profile Static Model by group
- References

2 Introduction

This specification defines a collection data types whose instances are used to represent the individual items of data from which *LRA* models are constructed. These data types therefore represent the finest levels of granularity to which data can be decomposed and each and every instance of data specified by an *LRA* model must conform to a specified data type.

The data types defined herein are designed to be re-usable across health care domains (i.e. domain independent) and are distinguished from higher-order, domain-specific abstractions by the facts that they assert no identity other than their value and that they are always immutable. Thus, two instances *x* and *y* of a data type, *D*, that have the same value, e.g. the number 5, are indistinguishable and completely interchangeable. An operation performed on instance *x* will yield the same result if it is performed on instance *y*. Data type instances are immutable by virtue of the fact that once created, any change of value implies a change of identity and thus the creation of a new instance.

2.1 Data Type Definitions

Within this specification each data type is described formally by a UML class definition. Each definition specifies a set of constraints that determines the allowable values of all possible instances of data that conform to the data type. A fuller description of these characteristics is described in ISO 21090.

2.2 Semantic Scope

It is the intention of this profile that:

the set of data types SHOULD match the semantic scopes of the set of *HL7 v3 R1* data types used in *MIM 7.2.02* and of the currently archetyped *openEHR* data types; and each data type SHOULD be expressive enough to satisfy the most common uses cases of the data type's instances.

This specification does not at present adopt generic collection types equivalent to the *HL7 v3 SET<T>* and *LIST<T>* data types. The semantics of simple collections are represented adequately using cardinality constraints with upper bound > 1 (e.g. [0..2], [0..*], etc), appended, if required, with an {ordered} constraint to indicate a

sequence. However, it is likely this may need to be revisited in order to model generated and sampled sequences derived from continuous monitoring.

2.3 ISO Notation

Data types in ISO 21090 are described using short uppercase strings (like TS - which is timestamp). These data types are constrained within the model to particular "flavours". These are described as the base data type name followed by a camel case flavour description (e.g. TS.DateTime, which is further constrained to TS.DateTime.Full).

Certain data types are specified in the abstract dependent on a parameter to define their full meaning. Thus the IVL or interval data type defines an interval between two quantities, and the type of quantity must be supplied before that IVL is realisable. This is parameter substitution is represented in the LRA documentation as it is in HL7 as IVL<TS>. In the ISO 21090 documentation this is usually referred to using brackets instead - i.e. IVL(TS), however in the associated XML schema it is instantiated as IVL_TS. Within the LRA the IVL<TS> notation will be used.

The one exception to this is that the continuous set of timestamps QSET<TS> is also referred to in the model and ISO documentation as GTS - or general timing specification.

2.4 Data type abbreviations

- HXIT: history
- ANY: any data
- BL: boolean
- QTY: quantities
- URG: uncertain range (parameterised)
- IVL: interval (parameterised)
- QSET: continuous set (parameterised)
- ED: encapsulated data
- ST: string
- SC: string with code
- TS: time stamp
- PIVL: periodic interval
- EIVL: event triggered interval
- ADXP: address part
- AD: address
- ENXP: entity name part
- EN: entity name
- CS: coded simple item

- CD: Coded data
- CO: Coded ordinal
- INT: integer
- REAL: real number
- RTO: ratio
- MO: monetary quantity
- PQV: physical quantity value
- PQ: physical quantity (UCUM units)
- PQR: physical quantity (coded units)
- II: instance identifier
- TEL: telecommunication locator

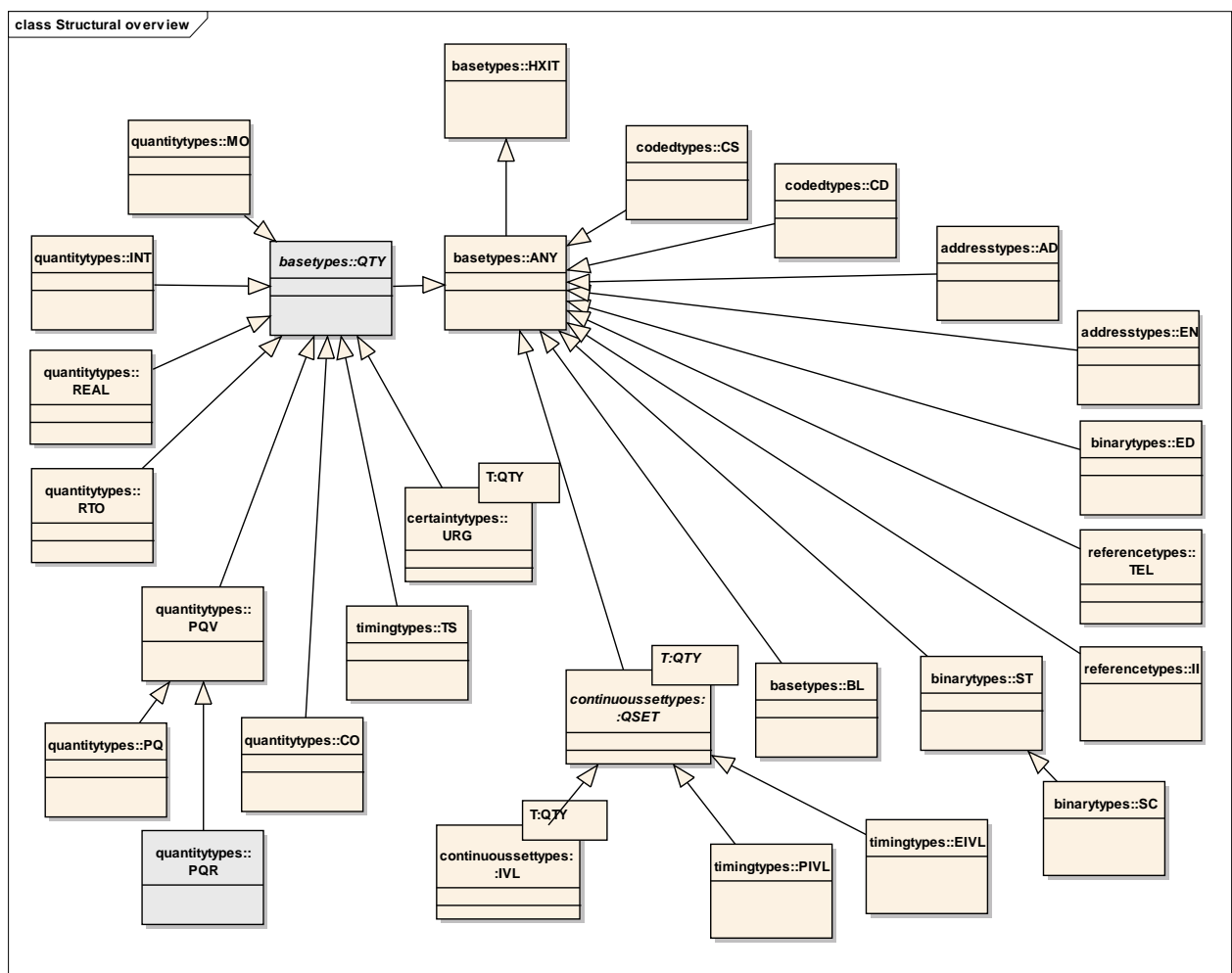


Figure 1: Structural overview: This diagram shows the key data types and their relationships to each other. Note that the different flavours of each type are not shown here.

3 Basic Data Types

Basic data types that provide infrastructural support for specific data types that are defined in subsequent sections.

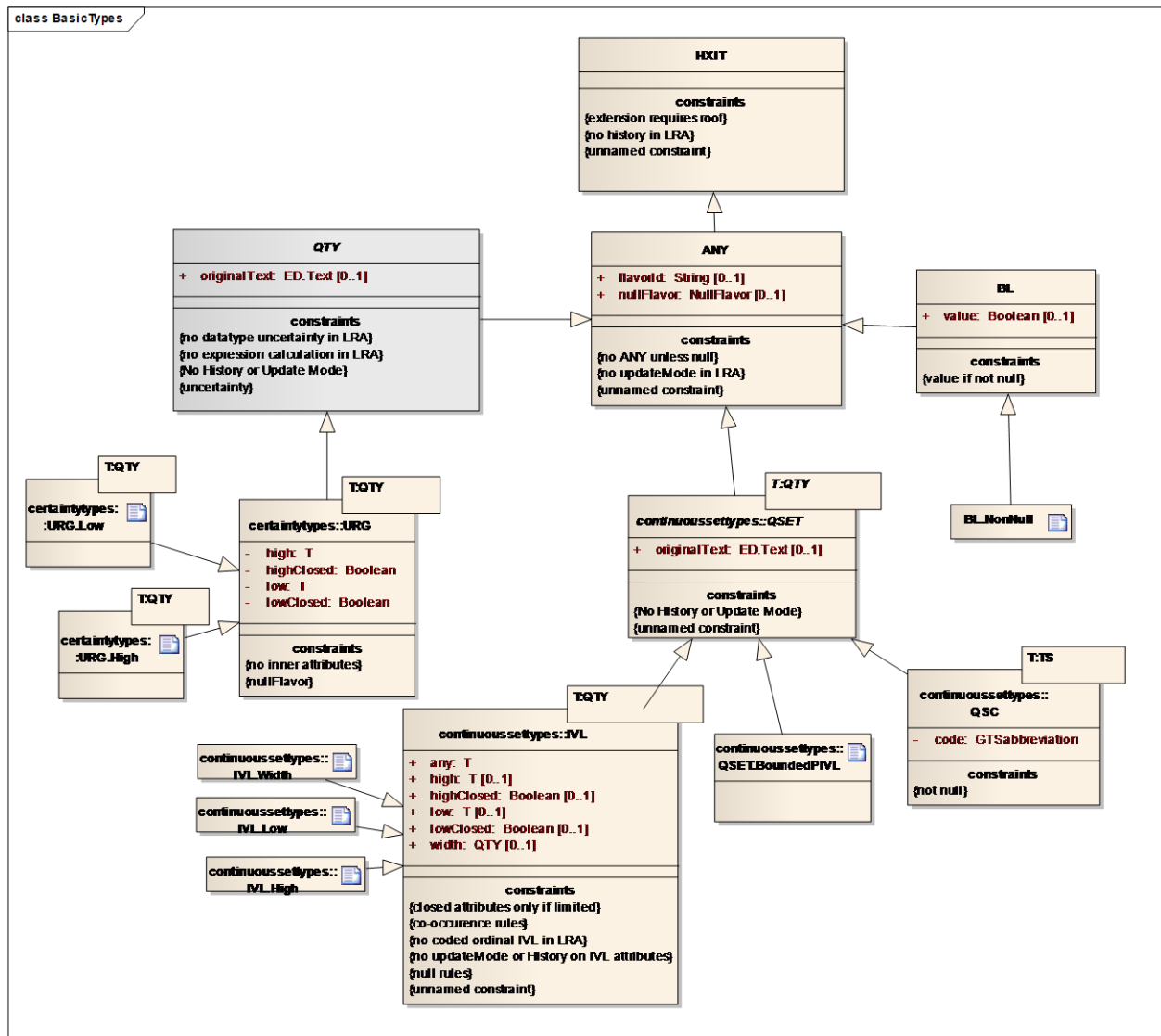


Figure 2: Basic Types

3.1 Package `Ira.datatypes.ISO 21090`

3.1.1 Class **HXIT** (history)

Specialises:

Realises:

This class is unused in the LRA. Within the LRA it is decided to not maintain history at a data type level, as it is felt that data type history potentially conflicts with the temporal context of a clinical entry, and the specific rules to guide usage could not be established.

Attributes

Attribute	Description
controlActExtension : String [0..1]	Is deprecated in the LRA. It MUST be undefined. Logic within the LRA MUST NOT depend on the value of controlActExtension
controlActRoot : Uid [0..1]	Is deprecated in the LRA. It MUST be undefined. Logic within the LRA MUST NOT depend on the value of controlActRoot.
validTimeHigh : String [0..1]	Is deprecated in the LRA. It MUST be undefined. Logic within the LRA MUST NOT depend on the value of validTimeHigh.
validTimeLow : String [0..1]	Is deprecated in the LRA. It MUST be undefined. Logic within the LRA MUST NOT depend on the value of validTimeLow.

Constraints

Constraint Type	Name	Details
LRA Invariant	no history in LRA	inv: validTimeLow.ocllsUndefined and validTimeHigh.ocllsUndefined and controlActRoot.ocllsUndefined and controlActExtension.ocllsUndefined
Definitional		def: noHistory : Boolean = validTimeLow.ocllsUndefined and validTimeHigh.ocllsUndefined and controlActRoot.ocllsUndefined and controlActExtension.ocllsUndefined
Invariant	extension requires root	inv: controlActExtension.ocllsDefined implies controlActRoot.ocllsDefined

3.1.2 Class ANY (any data)

Specialises: HXIT

Realises:

This abstract class is the root of all data types. The LRA data types profile does not allow for alteration of a value at the level of the data type therefore the ANY class is constrained to not require an update mode. This versioning capability is part of EN13606 and including it here would be inadvisable.

The collection classes (COLL, BAG, SET etc..) are out of scope of the data-types in the LRA. It is a specific information model construct in 13606 to define the behaviour of sets of data values and aggregations of those sets. In the first instance the LRA will mandate only core UML collection types, List, Set and Bag for specific

applications within the reference model, beyond that the structural elements provided by EN13606 are to be used.

Attributes

Attribute	Description
flavorId : String [0..1]	This is implemented as described in ISO 21090.
nullFlavor : NullFlavor [0..1]	This is implemented as described in ISO 21090.
updateMode : UpdateMode [0..1]	Is deprecated in the LRA. It is optional within ISO 21090 and MUST be undefined within the LRA. Logic within the LRA MUST NOT depend on the value of updateMode.

Constraints

Constraint Type	Name	Details
LRA Invariant	no updateMode in LRA	inv: updateMode.ocllsUndefined
Invariant	no ANY unless null	inv: (isNotNull or invalid) implies not ocllsTypeOf("ANY")
Definitional		def: isNull : Boolean = nullFlavor.ocllsDefined

3.1.3 Class BL (Boolean)

Specialises: ANY

Realises:

This is implemented as described in ISO 21090.

Attributes

Attribute	Description
value : Boolean [0..1]	This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	value if not null	inv: isNotNull implies value.ocllsDefined

3.1.4 Class BL.NonNull (Boolean – non null)

Specialises: BL

Realises:

This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	BL.NONNULL cannot be null	inv: not isNull

4 Text and Binary Data Types

Data types that provide support for text and multimedia data.

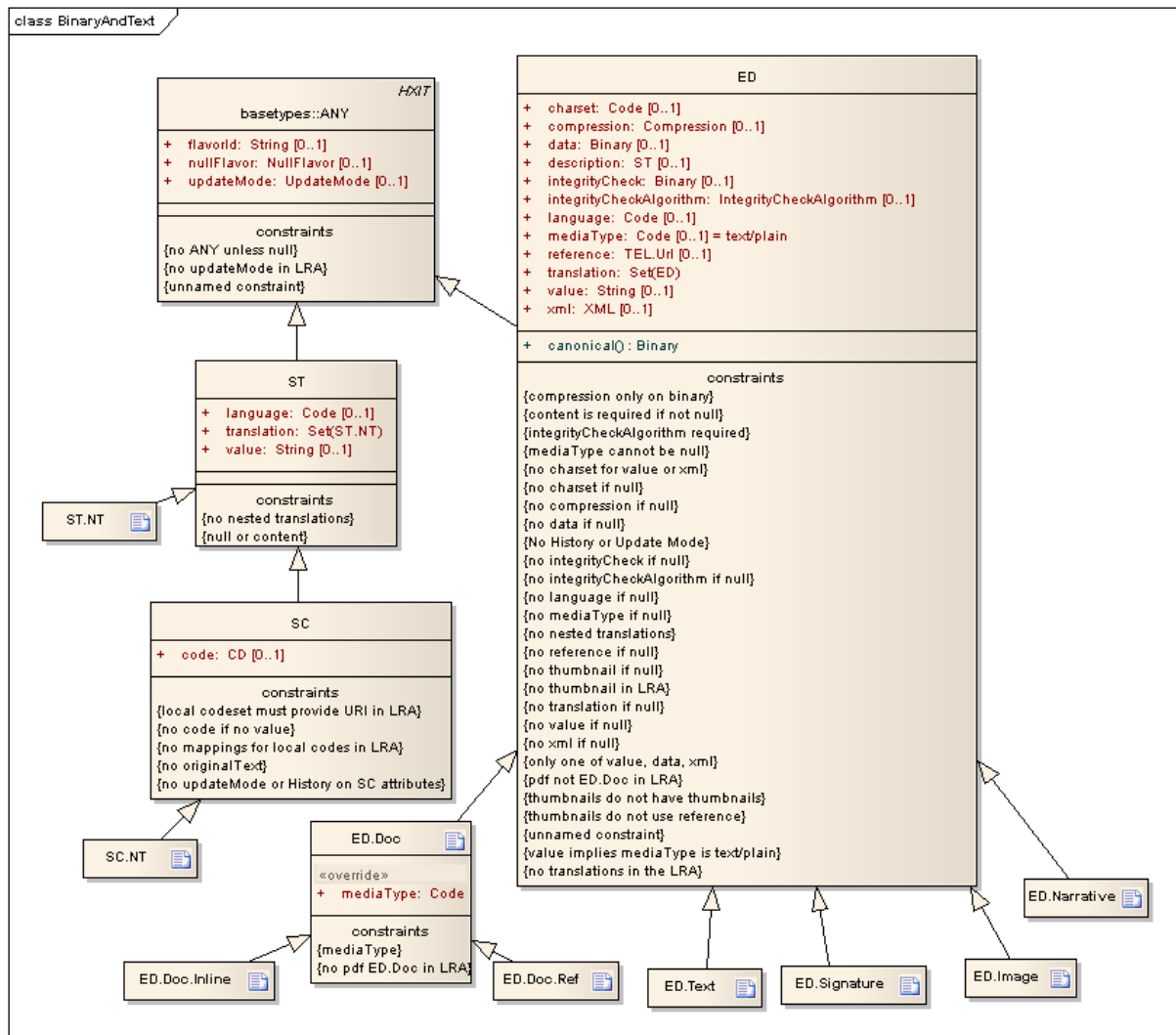


Figure 3: Binary And Text

4.1 Package Ira.datatypes.ISO 21090

4.1.1 Class ED (Encapsulated data)

Specialises: ANY

Realises:

This class is the abstract generalisation of classes whose instances are used to model data that has internal structure defined outside of the scope of the LRA.

Encapsulated data includes binary objects such as word processed documents and multimedia data as well as plain and marked-up text intended for human interpretation or machine processing outside the scope of the LRA.

Attributes

Attribute	Description
charset : Code [0..1]	in ISO 21090 this is defined as the charset of the encapsulation of data and not of the data itself. Data that can define its own charset (such as binary encoded XML for example) may do so. This attribute should not be interpreted as the charset of the canonical form of the data, which will be derived from its canonical content based on the underlying media type. This attribute is implemented as described in ISO 21090
compression : Compression [0..1]	This is implemented as described in ISO 21090.
data : Binary [0..1]	This is implemented as described in ISO 21090.
description : ST [0..1]	Is optional in the LRA. It is implemented within ISO 21090 as an American disability act requirement and MAY be undefined within the LRA. Logic within the LRA MUST NOT depend on the value of description.
integrityCheck : Binary [0..1]	This is implemented as described in ISO 21090.
integrityCheckAlgorithm : IntegrityCheckAlgorithm [0..1]	This is implemented as described in ISO 21090.
language : Code [0..1]	This is implemented as described in ISO 21090.
mediaType : Code [0..1]	This is implemented as described in ISO 21090. MediaType is described as mandatory if not null in 21090. This is modelled in 21090 and here as an optional attribute and a OCL constraint - "mediaType cannot be null"
reference : TEL.Url [0..1]	This is implemented as described in ISO 21090.
thumbnail : ED [0..1]	Is deprecated in the LRA. It is optional within ISO 21090 and MUST be undefined within the LRA. Logic within the LRA MUST NOT depend on the value of thumbnail.
translation : Set(ED) [1..1]	This attribute is deprecated in the LRA. It is optional within ISO 21090 in that the SET<ED> may be empty. It must be undefined within the LRA. Logic within the LRA MUST NOT depend on the value of translation.
value : String [0..1]	This is implemented as described in ISO 21090.
xml : XML [0..1]	In the instance of primary XML (or XHTML) data the ED representation is an encapsulated encoding of the XML, which may have a different character set to the original. The actual canonical data xml may be rendered inline in the ED.xml, but might not. If not, then the canonical representation may have to be derived to obtain the actual uncompressed xml. This is implemented as described in ISO 21090

Constraints

Constraint Type	Name	Details
LRA Invariant	no thumbnail in LRA	inv: thumbnail.ocllsUndefined

Constraint Type	Name	Details
LRA Invariant	pdf not ED.Doc in LRA	inv: mediaType = "application/pdf" implies flavourID.ocllsUndefined
LRA Invariant	no translation in the LRA	inv: translation->isEmpty
Invariant	no translation if null	inv: isNull implies translation->isEmpty
Invariant	value implies mediaType is text/plain	inv: value.ocllsDefined implies mediaType = 'text/plain'
Definitional		def: hasValue : Boolean = value.ocllsDefined or data.ocllsDefined or xml.ocllsDefined or hasReference
Invariant	only one of value, data, xml	inv: value.ocllsDefined implies (data.ocllsUndefined and xml.ocllsUndefined) and data.ocllsDefined implies (value.ocllsUndefined and xml.ocllsUndefined) and xml.ocllsDefined implies (value.ocllsUndefined and data.ocllsUndefined)
Invariant	thumbnails do not have thumbnails	inv: thumbnail.isNotNull implies thumbnail.thumbnail.ocllsUndefined
Invariant	no value if null	inv: isNull implies value.ocllsUndefined
Invariant	no xml if null	inv: isNull implies xml.ocllsUndefined
Invariant	thumbnails do not use reference	inv: thumbnail.isNotNull implies not thumbnail.hasReference
Invariant	no data if null	inv: isNull implies data.ocllsUndefined
Invariant	no integrityCheck if null	inv: isNull implies integrityCheck.ocllsUndefined
Invariant	integrityCheckAlgorithm required	inv: integrityCheck.ocllsDefined implies integrityCheckAlgorithm.ocllsDefined
Invariant	no reference if null	inv: isNull implies reference.ocllsUndefined
Invariant	no nested translations	inv: translation->forAll(t t.translation->isEmpty) -- nullFlavor invariants
Invariant	no mediaType if null	inv: isNull implies mediaType.ocllsUndefined
Invariant	no language if null	inv: isNull implies language.ocllsUndefined

Constraint Type	Name	Details
Invariant	no integrityCheckAlgorithm if null	inv: isNull implies integrityCheckAlgorithm.ocllsUndefined
Invariant	No History or Update Mode	inv: noUpdateOrHistory(reference) and noUpdateOrHistory(thumbnail)
Invariant	no compression if null	inv: isNull implies compression.ocllsUndefined
Invariant	no charset if null	inv: isNull implies charset.ocllsUndefined
Invariant	mediaType cannot be null	inv: isNotNull implies mediaType.ocllsDefined
Invariant	content is required if not null	inv: isNotNull implies hasValue
Invariant	compression only on binary	inv: compression.ocllsDefined implies data.ocllsDefined
Invariant	no thumbnail if null	inv: isNull implies thumbnail.ocllsUndefined
Invariant	no charset for value or xml	inv: (value.ocllsDefined or xml.ocllsDefined) implies charset.ocllsUndefined

4.1.2 Class ED.Doc (Encapsulated data – documentation)

Specialises: ED

Realises:

Instances of this class are used to model plain and marked-up text intended for human interpretation or machine processing outside the scope of the LRA. The choice of use of data type ED.Doc rather than ED.StructuredText is based on the nature of structured text documents in the LRA. At the data type level documents in the scope of the LRA are anticipated to be fragments or summaries of care that contain aspects that are machine processable rather than rich multimedia text. The complex specification of a whole document at a data type level is therefore not encouraged beyond the use of fragments of standard XHTML within an ED.Doc entity (the class specialisations ED.StructuredText and ED.StructuredTitle and associated StructuredText classes are out of scope).

Attributes

Attribute	Description
mediaType : Code [1..1]	In ISO 21090 ED.DOC is capable of representing plain text, html, xml or pdf documents. In the LRA pdf documents are disallowed for this specific specialisation type (although may be still represented using an unspecialised ED). Within the scope of the LRA this attribute MUST NOT contain "application/pdf". In the instance data from outside of the LRA is received that uses "application/pdf", the conforming application MUST treat the data as a vanilla EN data type.

Attribute	Description

Constraints

Constraint Type	Name	Details
LRA Invariant	no pdf ED.Doc in LRA	inv: mediaType = "text/plain" or mediaType = "text/html" or mediaType = "text/xml"
Invariant	mediaType	inv: mediaType = "text/plain" or mediaType = "text/html" or mediaType = "text/xml" or mediaType = "application/pdf"

4.1.3 Class ED.Doc.Inline (Encapsulated data – documentation inline)

Specialises: ED.Doc

Realises:

This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	no reference	inv: reference.ocllsUndefined

4.1.4 Class ED.Doc.Ref (Encapsulated data – documentation by reference)

Specialises: ED.Doc

Realises:

This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	reference required	inv: isNotNull implies reference.isNotNull

4.1.5 Class ED.Image (Encapsulated data – image)

Specialises: ED

Realises:

This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	no text	inv: value.ocllsUndefined
Invariant	fixed to image	inv: mediaType.substring(0,6) = "image/"
Invariant	no xml	inv: xml.ocllsUndefined

4.1.6 Class ED.Signature (Encapsulated data – electronic signature)

Specialises: ED

Realises:

This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	no compression	inv: compression.ocllsUndefined
Invariant	no language	inv: language.ocllsUndefined
Invariant	no value	inv: value.ocllsUndefined
Invariant	no translations	inv: translation->isEmpty
Invariant	no reference	inv: reference.ocllsUndefined
Invariant	no integrityCheck	inv: integrityCheck.ocllsUndefined
Invariant	no data	inv: data.ocllsUndefined
Invariant	mediaType	inv: mediaType = "text/xml"
Invariant	no thumbnail	inv: thumbnail.ocllsUndefined

4.1.7 Class ED.Text (Encapsulated data – text)

Specialises: ED

Realises:

The specialised class ED.Text overlaps the function of ED.Doc, and differs principally in that it is not allowed to be compressed or translated. In the LRA there is no distinction drawn between the two classes, ED.Text is regarded as a subtype of

ED.Doc, a conforming application MAY treat all ED.Text as it would treat ED.Doc.
This is implemented as described in ISO 21090

Constraints

Constraint Type	Name	Details
Invariant	no data	inv: data.ocllsUndefined
Invariant	no compression	inv: compression.ocllsUndefined
Invariant	no translations	inv: translation->isEmpty
Invariant	no integrityCheck	inv: integrityCheck.ocllsUndefined
Invariant	no thumbnail	inv: thumbnail.ocllsUndefined
Invariant	text only	inv: mediaType = "text/plain"
Invariant	no xml	inv: xml.ocllsUndefined

4.1.8 Class ST (String)

Specialises: ANY

Realises:

This is implemented as described in ISO 21090.

Attributes

Attribute	Description
language : Code [0..1]	This is implemented as described in ISO 21090.
translation : Set(ST.NT) [1..1]	This is implemented as described in ISO 21090.
value : String [0..1]	This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	null or content	inv: isNull xor (value.ocllsDefined and value.size > 0)
Invariant	no nested translations	inv: translation->forAll(t t.translation->isEmpty)

4.1.9 Class ST.NT (String – no translations)

Specialises: ST

Realises:

This is implemented as described in ISO 21090.

Relationships

Relationship Type	Source	Target	Description
Generalization	ST.NT	ST	

Constraints

Constraint Type	Name	Details
Invariant	no translations	inv: translation->isEmpty

4.1.10 Class SC (String with code)

Specialises: ST

Realises:

In ISO 21090 the SC class is closely related to the CD class. Differences are highlighted there, but within the LRA the usage of the SC class is restricted to circumstances where a LRA conformant system sends information that is part of an enumerated list, where that enumeration is local to the sending system. The enumeration **MUST** be represented by the controlled string held in the value attribute and **MAY** be associated with a code from a local (to the sending system) coding scheme. The SC class **MUST NOT** be used to convey coded information using SNOMED CT, or any other globally controlled code set. Logic within the LRA **MUST NOT** rely on the coded value of SC. The SC.code attribute **MUST NOT** be used to represent document type coding, as such document strings **MUST** be represented as ED.DOC. Document type encoding is not performed at data type level.

The use of SC as a criterion for conformance within the LRA specification is to be avoided, its use is optional within the LRA as an implementation alternative for ST where a system supplier deems it useful for internal processes. It may possibly be used so a conformant system **MUST** be able to interpret the SC.value attribute of the data type.

This is implemented as described in ISO 21090.

Attributes

Attribute	Description
code : CD [0..1]	In the use of the SC datatype as described above there are specific constraints that must be applied to the CD value of the SC.code attribute. In ISO 21090 there is already a constraint to have no originalText attribute. In the LRA the SC.code MUST not contain source mappings, the SC.code.codeSystemName MUST provide a identifier as a string consisting of a URL to a maintained accessible terminology service. The SC.code.codeSystemVersion and SC.code.code strings must act as extensions to that URL to derive the specific value as specified in SC.Value (e.g.

Attribute	Description
	<p>SC.code.codeSystemName = “http://some.ehr.manufacturer/vocabserver/somevocabulary” SC.code.codeSystemVersion = “v1”, SC.code.code = “Code1234”. Then the URI “http://some.ehr.manufacturer/vocabserver/somevocabulary/v1#Code1234” MUST resolve to SC.Value. - see http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/#section-URI-Vocabulary for further examples.). At a most basic level this can be a stable HTML page which defines anchored elements for every entry in the local terminology.</p>

Constraints

Constraint Type	Name	Details
LRA Invariant	local codeset must provide URI in LRA	inv: code.codeSystemName.concat(code.codeSystemVersion.concat('#.concat(code.code))).matches("[a-zA-Z][0-9a-zA-Z+\\-\\.]*.*/{0,2}[0-9a-zA-Z;/?:@&=+\$\\-!~*()'%%]+)?(#[0-9a-zA-Z;/?:@&=+\$\\-!~*()'%%]+)?")
LRA Invariant	no mappings for local codes in LRA	inv:code.isNotNull implies code.source.oclIsUndefined
Invariant	no updateMode or History on SC attributes	inv: noUpdateOrHistory(code)
Invariant	no originalText	inv: code.isNotNull implies code.originalText.isNotNull
Invariant	no code if no value	inv: code.isNotNull implies isNotNull

4.1.11 Class SC.NT (Coded string no translation)

Specialises: SC

Realises:

This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	no translations	inv: translation->isEmpty

5 Coded Datatypes (Terminology)

These data types provide support for use of codes and terms from terminologies and classifications. Important information regarding the specification of this section is held in Appendix B. It is recommended to read this section and appendix B together.



Figure 4: Coded Data

5.1 Package Ira.datatypes.ISO 21090

5.1.1 Class CS (Coded simple item)

Specialises: ANY

Realises:

This is implemented as described in ISO 21090.

Attributes

Attribute	Description
code : String [0..1]	This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	no code if null	inv: isNull implies code.ocllsUndefined
Invariant	code is required	inv: isNotNull implies code.ocllsDefined -- nullFlavor invariants

5.1.2 Class CD (Coded data)

Specialises: ANY

Realises:

Instances of this class are used to model coded data, which may take the form of either a simple enumerated coded concept (i.e. a code and display name pair) or a coded expression comprising any number of coded concepts.

The LRA approach to coded data is focussed on use of SNOMED CT. However SNOMED CT is not the only terminology system that is expected to be needed, and the CD data type needs to be capable of representing multiple coding systems.

An open issue with the use of CD is the ability to represent different 'complete' renderings of a post coordinated SNOMED CT expression based on the use cases. The problem with originalText is that it merges different types of rendering. One of those could be a 'generated rendering' and the other could be a literal 'original text typed' while a third would be some text associated with the expression displayed to the user for select. Within ISO 21090 this is not currently possible.

5.1.2.1

Attributes

Attribute	Description
code : String [0..1]	This attribute specifies the plain code symbol defined by the code system, or an expression in a syntax defined by the code system which describes the concept. This is a string representation of a code symbol or expression as defined by the coding system.
codeSystem : Uid [0..1]	This is implemented as described in ISO 21090.
codeSystemName : String [0..1]	This is implemented as described in ISO 21090.
codeSystemVersion : String [0..1]	This is implemented as described in ISO 21090.
codingRationale : Set(CodingRationale) [1..1]	The coding rationale attribute is optional in ISO 21090, and appears to be used to support translations (see below) It use is deprecated in the LRA.
displayName : ST [0..1]	The use of this attribute is optional within ISO 21090 and deprecated in the LRA.
originalText : ED.Text [0..1]	The originalText attribute is the primary textual representation of a coded string, and can either be provided inline or by reference. It refers to the human readable rendering of the concept which was the source of the assigned code (which may be post hoc in 21090). A separate requirement exists for a structured text rendering of

Attribute	Description
	<p>a code or code phrase within a number of defined contexts (e.g. verbose, terse, structured format) this attribute is not appropriate for this use. At present this use case is unmet in ISO 21090, and it may be that it may be best met by a specification of rules or through an operation to define the rendering pattern for individual contexts. .</p> <p>This is implemented as described in ISO 21090.</p>
source : CD [0..1]	<p>If a CD is a result of a translation or mapping, having been performed then the original CD element is represented as the source attribute. The LRA requires that this be supported, and that this is used to represent the source (or origin) of any code mapping. In the specific example of Read to SNOMED mapping source MUST contain the original read code. It is possible for a source to also have a source attribute. Thus an entire history of translation should be maintained for any translated or mapped code.</p> <p>.</p> <p>This is implemented as described in ISO 21090.</p>
translation : Set(CD) [1..1]	<p>In ISO21090 the assertion of equivalence to a concept in another coding system is enabled through a stated set of translations. In the LRA it is believed that the use cases for “before the event” code translation is not clear, and the potential for misuse high. This optional attribute in ISO 21090 is deprecated within the LRA and systems MUST NOT supply it. Logic in the LRA MUST NOT rely on this attribute. It is noted that language translations (in contrast to codeset translations) are not supported in the CD datatype “ language translations are a feature of some terminology systems.</p>
valueSet : String [0..1]	<p>This attribute is optional in ISO 21090 and deprecated in the LRA. ValueSets are a property of the information model slots and may be expressed not as sets but constraints.</p>
valueSetVersion : String [0..1]	<p>This attribute is optional in ISO 21090 and deprecated in the LRA. ValueSets are a property of the information model slots in archetyped 13606 models and may be expressed not as sets but terminology constraints in the LRA.</p>

5.1.2.2 relationships

Relationship Type	Source	Target	Description
Generalization	CD	ANY	
Generalization	CD.CV	CD	

5.1.2.3 constraints

Constraint Type	Name	Details
-----------------	------	---------

Constraint Type	Name	Details
Invariant	code requires codeSystem	inv: code.ocllsDefined implies codeSystem.ocllsDefined
Invariant	other requires codeSystem or valueSet	inv: (nullFlavor = NullFlavor.OTH) implies (codeSystem.ocllsDefined or valueSet.ocllsDefined)
Invariant	valueSet requires valueSetVersion	inv: valueSet.ocllsDefined implies (valueSetVersion.ocllsDefined)
Invariant	Translations cannot have translations	inv: translation->forAll(t t.translation->isEmpty)
Invariant	null or code and/or originalText	inv: isNotNull implies (hasCode or hasOriginalText)
Invariant	no updateMode or History on CD elements	inv: noUpdateOrHistory(displayName) and noUpdateOrHistory(originalText) and translation->forAll(t noUpdateOrHistory(t)) -- nullFlavor invariants
Invariant	no source if null	inv: isNull implies source.ocllsUndefined
Invariant	No original text on translations	inv: translation->forAll(t t.noOriginalText)
Invariant	no displayName if null	inv: isNull implies displayName.ocllsUndefined
Invariant	no code if null	inv: isNull implies code.ocllsUndefined
Invariant	displayName only if code	inv: displayName.ocllsDefined implies code.ocllsDefined
Invariant	codeSystemVersion only if codeSystem	inv: codeSystemVersion.ocllsDefined implies codeSystem.ocllsDefined
Invariant	codeSystemName only if codeSystem	inv: codeSystemName.ocllsDefined implies codeSystem.ocllsDefined
Invariant	unnamed constraint	def: hasCode : Boolean = code.ocllsDefined
LRA Invariant	no code translation in LRA	inv: translation.ocllsUndefined
LRA Invariant	no valueSet in LRA	inv: valueSet.ocllsUndefined and valueSetVersion.ocllsUndefined
LRA Invariant	no displayName in LRA	inv: displayName.ocllsUndefined
LRA Invariant	no codingRationale in LRA	inv: codingRationale.ocllsUndefined

Constraint Type	Name	Details
LRA Invariant	no originalText by reference in LRA	inv: originalText.reference.ocllsUndefined

5.1.3 Class CD.CV (Coded data – no translations)

Specialises: CD

Realises:

This is implemented as described in ISO 21090.

5.1.3.1 relationships

Relationship Type	Source	Target	Description
Generalization	CD.CV.SCT	CD.CV	
Generalization	CD.CV.SCT	CD.CV	
Generalization	CD.CV	CD	

5.1.3.2 constraints

Constraint Type	Name	Details
Invariant	no source	inv: source.ocllsUndefined
Invariant	no translations	inv: translation.isEmpty

5.1.4 Class CD.CV.SCT (SNOMED CT Coded data – no translations)

Specialises: CD.CV

Realises:

A constraint of type CD.CV in which the code specifies a single concept or a composite expression containing multiple concepts drawn from SNOMED CT.

5.1.4.1 *relationships*

Relationship Type	Source	Target	Description
Generalization	CD.CV.SCT	CD.CV	

5.1.4.2 *constraints*

Constraint Type	Name	Details
LRA Invariant	code is type SnomedCtExpression	inv: code.ocllsKindOf(SnomedCtExpression)

6 Identity and Location Datatypes

These data types provide support for identifying objects, records, and things, and specifically for URLs, URIs and telecommunication addresses.

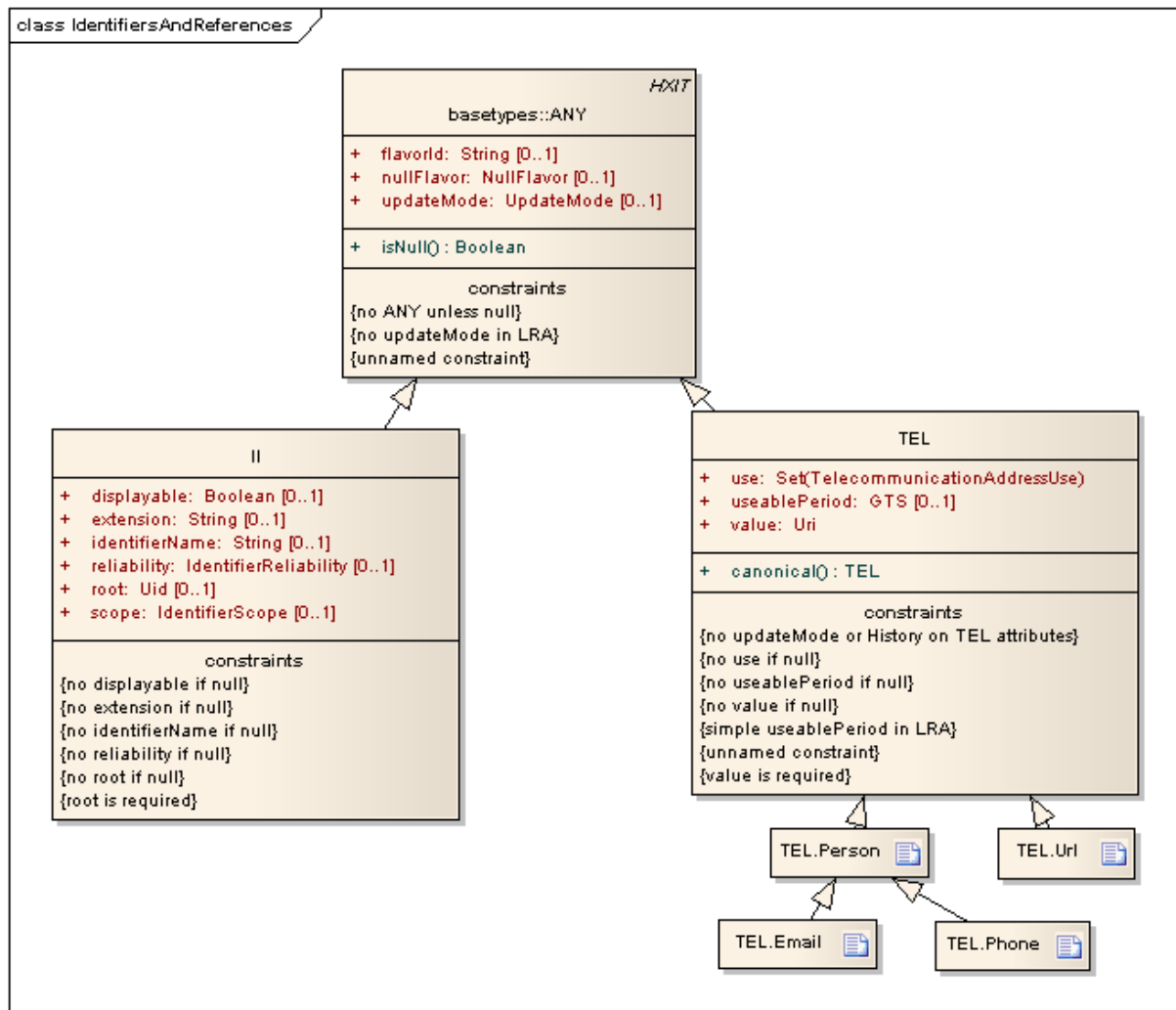


Figure 5: Identifiers And References

6.1 Package Ira.datatypes.ISO 21090

6.1.1 Class II (Instance identifier)

Specialises: ANY

Realises:

This is implemented as described in ISO 21090.

Attributes

Attribute	Description
displayable : Boolean [0..1]	This is implemented as described in ISO 21090.
extension : String [0..1]	This is implemented as described in ISO 21090.
identifierName : String [0..1]	This is implemented as described in ISO 21090.
reliability : IdentifierReliability [0..1]	This is implemented as described in ISO 21090.

Attribute	Description
root : Uid [0..1]	This is implemented as described in ISO 21090.
scope : IdentifierScope [0..1]	This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	root is required	inv: isNotNull implies root.ocllsDefined -- nullFlavor invariants
Invariant	no root if null	inv: isNull implies root.ocllsUndefined
Invariant	no reliability if null	inv: isNull implies reliability.ocllsUndefined
Invariant	no identifierName if null	inv: isNull implies identifierName.ocllsUndefined
Invariant	no extension if null	inv: isNull implies extension.ocllsUndefined
Invariant	no displayable if null	inv: isNull implies displayable.ocllsUndefined

6.1.2 Class TEL (Telecommunications locator)

Specialises: ANY

Realises:

The telecommunication address class in ISO 21090 is very similar to HL7's TEL class. As such the LRA will conform to the MiM's use of the HL7 data type for compatibility reasons.

Attributes

Attribute	Description
use : Set(TelecommunicationAddressUse) [1..1]	<p>The LRA shall use a constrained set of the vocabulary set as defined by the MiM, to maintain backward compatibility with the MiM. The following subset of the HL7 standard Telecom. addresses shall be used:</p> <p>"HP" - home</p> <p>"HV" - vacation or other temporary number)</p> <p>"AS" - an automatic answering machine</p> <p>"WP" - work</p> <p>"MC" - mobile number)</p> <p>"PG" - pager</p> <p>"EC" - emergency contact.</p> <p>This constraint is applied through the use of specific LRA vocabulary and is not formally expressed here.</p>

Attribute	Description
useablePeriod : GTS [0..1]	The LRA shall use an IVL<TS> constrained form of the ISO 21090 QSET<TS> (=GTS) value for this attribute to maintain backward compatibility with the MiM. This constraint may be relaxed in the future.
value : Uri [1..1]	This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
LRA Invariant	simple useablePeriod in LRA	inv: useablePeriod.ocllsTypeOf(IVL<TS>)
Invariant	no use if null	inv: isNull implies use->isEmpty
Invariant	no updateMode or History on TEL attributes	inv: noUpdateOrHistory(useablePeriod)
Invariant	no value if null	inv: isNull implies value.ocllsUndefined
Definitional		def: scheme(s : String) : Boolean = value.substring(0, s.size) = s -- nullflavor invariants:
Invariant	no useablePeriod if null	inv: isNull implies useablePeriod.ocllsUndefined
Invariant	value is required	inv: isNotNull implies value.ocllsDefined

6.1.3 Class TEL.Email (Email address)

Specialises: TEL.Person

Realises:

This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	email only	inv: scheme("mailto")

6.1.4 Class TEL.Person (Person telecommunication locator)

Specialises: TEL

Realises:

This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	Personal Address	inv: scheme("tel") or scheme("x-text-fax") or scheme("x-text-tel") or scheme("mailto")

6.1.5 Class TEL.Phone (Telephone number)*Specialises:* TEL.Person*Realises:*

This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	Phone	inv: scheme("tel") or scheme("x-text-fax") or scheme("x-text-tel")

6.1.6 Class TEL.Url (Universal resource locator)*Specialises:* TEL*Realises:*

This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	schemes	inv: (scheme("file") or scheme("ftp") or scheme("cid") or scheme("http") or scheme("https") or scheme("nfs"))
Invariant	no use	inv: use->isEmpty

7 Name and Address Datatypes

These data types provide support for names and addresses.

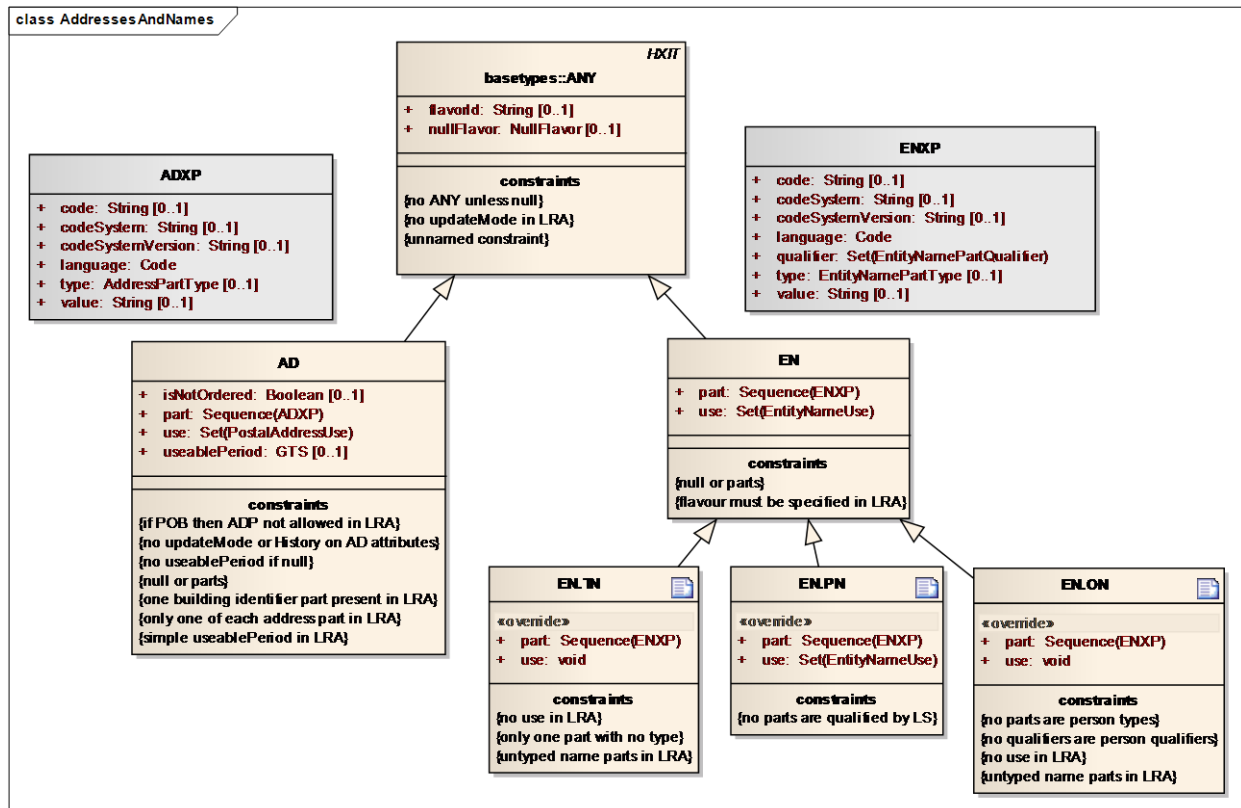


Figure 6: Addresses And Names

7.1 Package Ira.datatypes.ISO 21090

7.1.1 Class ADXP (Address part)

Specialises:

Realises:

This defines the parts of an address. This behaves in an identical way to the HL7 address class and hence the LRA constrains it in an identical way to the MiM.

Attributes

Attribute	Description
code : String [0..1]	This is implemented as described in ISO 21090.
codeSystem : String [0..1]	This is implemented as described in ISO 21090.
codeSystemVersion : String [0..1]	This is implemented as described in ISO 21090.
language : Code [1..1]	This is implemented as described in ISO 21090.
type : AddressPartType [0..1]	In ISO 21090 the address part type nomenclature is specified as the HL7 AddressPartType code system. In the LRA this must be constrained to fields that map to the postal address file from the royal mail as is currently the case in the MiM. The type shall be one of the following, which replace the HL7 standard types for UK use:

Attribute	Description
	<p>"POB" - post box number</p> <p>"ADP" - address prefix, e.g. location within a building or complex, such as "Flat 22". Similar to HL7 "UNID"</p> <p>"BI" - building identifier, e.g. house name or number". Broader version of HL7 "BNR"</p> <p>"STR" - street or road name. HL7 standard</p> <p>"LCY" - locality. A refinement of HL7 "CTY"</p> <p>"TN" - town. A refinement of HL7 "CTY"</p> <p>"CPA" - county. HL7 standard</p> <p>"CNT" - country. HL7 standard</p> <p>In using the ISO 21090 data types however the ability to use an untyped string to represent the unstructured address line has been deprecated. It is possible to use an untyped ADXP (rather than a string) to represent entries that are represented in the HL7 unstructured address flavours, and this untyped ADXP is included in the LRA profile for the specific purpose of backward support.</p> <p>Allowable types are therefore; POB, ADP, BI, STR, LCY, TN, CPA, CNT</p> <p>Type is not mandatory. This constraint is applied through the use of specific LRA vocabulary and is not formally expressed here.</p> <p>

</p>
value : String [0..1]	This is implemented as described in ISO 21090.

7.1.2 Class AD (Address)

Specialises: ANY

Realises:

In the Npfit MiM Datatypes implementation manual (section 2.4.4 version 3.6) there is a full description of the allowable patterns of HL7 addresses as supported by Npfit. The form of ISO 21090 addresses inherits many aspects from HL7, although the unstructured form of address is deprecated. The LRA profile hence will adopt the same patterns of use as the MiM flavour of "address fully structured with coded country"

For details please see the MiM V7.02. Only one of each type (other than a delimiter) may occur in an address, and one "buildingIdentifier" part shall be present. If "poBox" is used, "address prefix" may not be used.

Attributes

Attribute	Description
isNotOrdered : Boolean [0..1]	This is implemented as described in ISO 21090.
part : Sequence(ADXP) [1..1]	This is implemented as described in ISO 21090.
use : Set(PostalAddressUse) [1..1]	<p>HL7 standard address use types supported by NPfit:</p> <p>"H" - Home address</p> <p>"HP" - Primary home</p>

Attribute	Description
	<p>"TMP" -Temporary address</p> <p>"PST" - Correspondence address</p> <p>"WP" - Work</p> <p>This constraint is applied through the use of specific LRA vocabulary and is formally expressed here.</p>
useablePeriod : GTS [0..1]	in ISO 21090 this attribute has been changed from HL7's IVL<TS> to a general timing specification QSET<TS> (=GTS) whilst there is a use case for a GTS specification, from a pragmatic approach as interfaces have been developed against the MiM it seems not sensible to demand a different degree of conformance, therefore in the LRA at present it is recommended to constrain this attribute to be IVL<TS> as is currently the case in the MiM. This is forward compatible with adoption of a GTS.

Constraints

Constraint Type	Name	Details
LRA Invariant	simple useablePeriod in LRA	inv: useablePeriod.ocllsTypeOf(IVL<TS>)
LRA Invariant	one building identifier part present in LRA	inv: part.one(type = "BI")
LRA Invariant	only one of each address part in LRA	inv: part->IsUnique(type)
LRA Invariant	if POB then ADP not allowed in LRA	inv: part->exists(type = "POB") implies part->collect(type = "ADP")->count() = 0
Invariant	null or parts	inv: isNull xor parts->notEmpty -- nullflavor invariants:
Invariant	no useablePeriod if null	inv: isNull implies (useablePeriod.ocllsUndefined or useablePeriod.isNull)
Invariant	no updateMode or History on AD attributes	inv: noUpdateOrHistory(useablePeriod)

7.1.3 Class ENXP (Entity name part)

Specialises:

Realises:

This is a typed part of an entity name. The typing is an optional attribute and an untyped ENXP is essentially a string value. This untyped flavour of ENXP is used for organisational names, trivial names and for legacy support in people names.

Attributes

Attribute	Description
-----------	-------------

Attribute	Description
code : String [0..1]	This is implemented as described in ISO 21090.
codeSystem : String [0..1]	This is implemented as described in ISO 21090.
codeSystemVersion : String [0..1]	This is implemented as described in ISO 21090.
language : Code [1..1]	This is implemented as described in ISO 21090.
qualifier : Set(EntityNamePartQualifier) [1..1]	This is implemented as described in ISO 21090.
type : EntityNamePartType [0..1]	In ISO 21090 although ENXP is defined as a component part of EN.ON (organisational name) and EN.TN (trivial name), within the LRA the type attribute is only permitted for EN.PN and hence only describe aspects of a person name. The 5 allowed values (family, given, prefix, suffix, delimiter) are the same as in HL7 and all are supported by the LRA in the context of an EN.PN. No change of use over HL7's PN as defined in the MiM is required.
value : String [0..1]	This is implemented as described in ISO 21090.

7.1.4 Class EN (Entity name)

Specialises: ANY

Realises:

Entity names include names of organisations people and things. The ISO 21090 data type provides support for valid periods for such entity names to be specified. The NpflIT MiM implements the related HL7 datatypes PN (person name) ON (organisation name) and TN (trivial name for places and things) which map cleanly to the EN constraints EN.PN, EN.ON and EN.TN in ISO 21090. The MiM defines several flavours of entity name, but the ISO 21090 data types preclude the ability to use unstructured text as an entity name, however it is possible to use an untyped ENXP as part of a name to represent a plain string. Therefore the LRA profile will adopt the same implementation constraints as the MiM does for ON, EN and the fully structured flavour of PN as defined in MiM 7.02 Datatypes implementation manual v3.6. EN is not abstract in ISO 21090, but in the LRA profile its use is restricted to only its constrained subtypes, EN.PN (people), EN.ON (organisations) and EN.TN (anything else).

Attributes

Attribute	Description
part : Sequence(ENXP) [1..1]	This is implemented as described in ISO 21090.
use : Set(EntityNameUse) [1..1]	The adoption of the use attribute in the LRA is dependent on which constraint is applied. See individuals for details.

Constraints

Constraint Type	Name	Details
Invariant	null or parts	inv: isNull xor part->isEmpty
Invariant	flavour must be specified in LRA	inv: flavourId.ocllsDefined

7.1.5 Class EN.ON (Organisation Name)

Specialises: EN

Realises:

Organisational names in the LRA profile are not typed according to use, nor are their component parts typed. This maintains a backward compatibility with the MiM use of ON.

Attributes

Attribute	Description
part : Sequence(ENXP) [1..1]	In the context of the EN.ON in the LRA the ENXP types provided must all be untyped.
use : void [1..1]	in the LRA there is no defined vocabulary of different types of organisational name and this attribute is deprecated in the context of EN.ON in the LRA.

Constraints

Constraint Type	Name	Details
LRA Invariant	untyped name parts in LRA	inv: part.type.ocllsUndefined
LRA Invariant	no use in LRA	inv: use.ocllsUndefined
Invariant	no qualifiers are person qualifiers	inv: part->forAll(p not (p.type = EntityNamePartQualifier.I or p.type = EntityNamePartQualifier.P or p.type = EntityNamePartQualifier.ANON or p.type = EntityNamePartQualifier.A or p.type = EntityNamePartQualifier.R or p.type = EntityNamePartQualifier.DN or p.type = EntityNamePartQualifier.M))
Invariant	no parts are person types	inv: part->forAll(p not (p.type = EntityNamePartType.FAM or p.type = EntityNamePartType.GIV))

7.1.6 Class EN.PN (Person Name)

Specialises: EN

Realises:

This differs to the HL7 PN class as it cannot contain a simple string for unstructured names but instead it can include an untyped ENXP instance to support the unstructured flavours of EN as defined in the MiM.

Attributes

Attribute	Description
part : Sequence(ENXP) [1..1]	This is implemented as described in ISO 21090.
use : Set(EntityNameUse) [1..1]	<p>In the specific context of EN.PN, as with the MiM implementation of the HL7 PN class, only a subset of the full range of standard name types is supported by NpflIT and hence the LRA implementation of ISO 21090.</p> <p>"L" - Usual (current) name</p> <p>"A" - Alias name</p> <p>NpflIT additions to the set of ISO 21090 standard name types are:</p> <p>"PREFERRED" - Preferred name</p> <p>"PREVIOUS-BIRTH" - Birth name</p> <p>"PREVIOUS-BACHELOR" - Bachelor name</p> <p>"PREVIOUS-MAIDEN" - Maiden name</p> <p>"PREVIOUS" - Other previous name</p> <p>This constraint is applied through the use of specific LRA vocabulary and is not formally expressed here.</p>

Constraints

Constraint Type	Name	Details
Invariant	no parts are qualified by LS	inv: part->forAll(p not p.qualifier->includes(EntityNamePartQualifier.LS))

7.1.7 Class EN.TN (Trivial Name)

Specialises: EN

Realises:

Trivial names in the LRA profile are not typed according to use, nor are their component parts typed. This maintains a backward compatibility with the MiM use of TN in the LRA.

Attributes

Attribute	Description
part : Sequence(ENXP) [1..1]	In the context of the EN.ON in the LRA the ENXP types provided must all be untyped.
use : void [1..1]	in the LRA there is no defined vocabulary of different types of organisational name and this attribute is deprecated in the context of EN.ON.

Constraints

Constraint Type	Name	Details
LRA Invariant	untyped name parts in LRA	inv: part.type.ocllsUndefined
LRA Invariant	no use in LRA	inv: use.ocllsUndefined
Invariant	only one part with no type	inv: isNotNull implies (part->size = 1 and part->first.type.ocllsUndefined and part->first.qualifier.isNotNull

8 Package Ira.datatypes.ISO 21090.quantitytypes

These data types provide support for Quantitative values

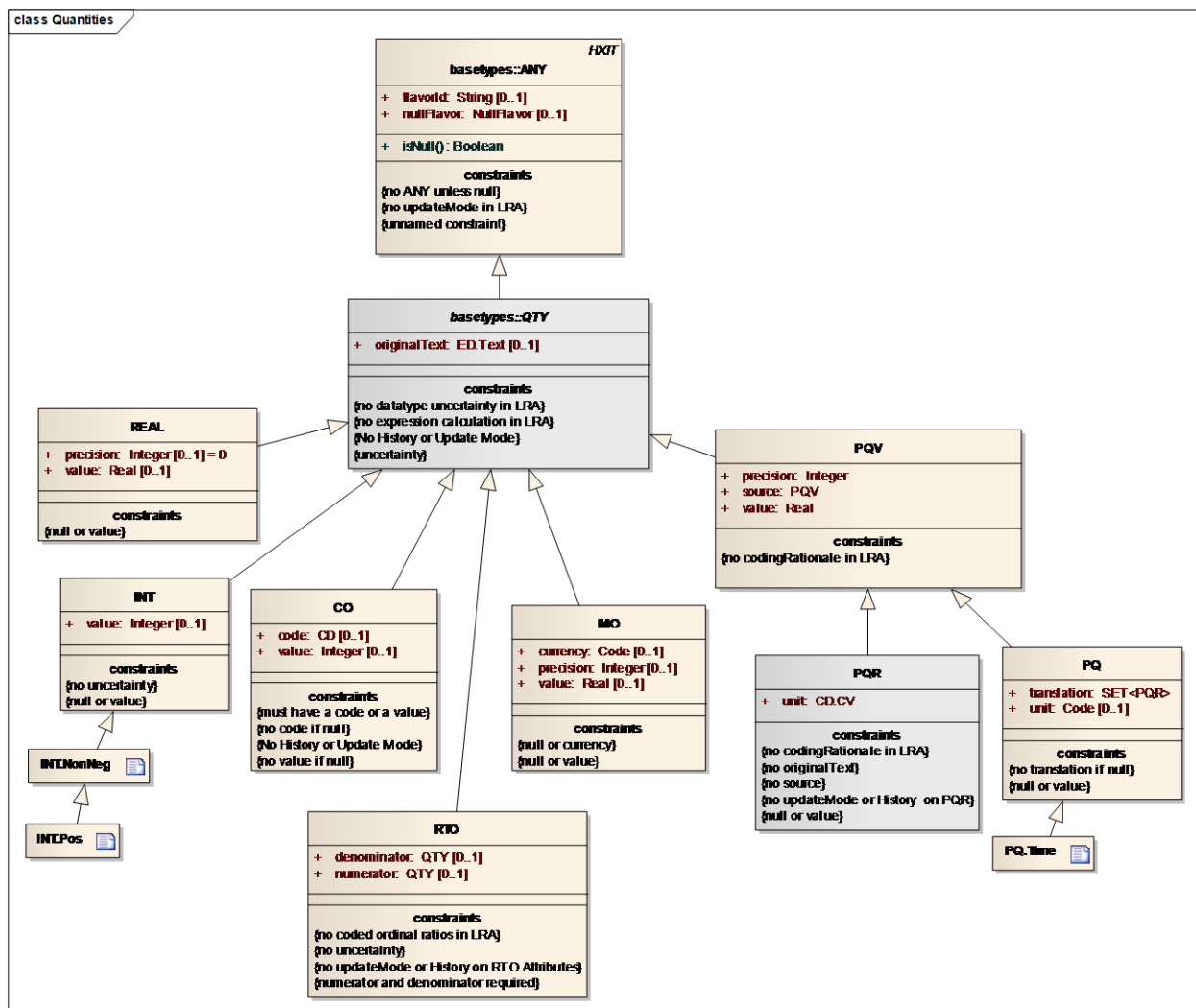


Figure 7: Quantities

8.1 Package Ira.datatypes.ISO 21090

8.1.1 Abstract Class QTY (quantity)

Specialises: ANY

Realises:

This class is the abstract generalization of all classes whose instances are used to model quantities. All quantities are numeric such that (a) the order of the numbers reflects an order relation (less-or-equal) defined for the quantified attribute and (b) the differences between the numbers reflect differences of the attribute.

Attributes

Attribute	Description
expression : ED [0..1]	in ISO 21090 the expression attribute allows for a specification of a derived value given contextual information, using an expression language such as OCL. It is intended in the LRA to make such derived values explicitly stated in terms of queries against a logical model. Use of this attribute is deprecated in the LRA.
expressionLanguage : Code [0..1]	This is deprecated in the LRA.
originalText : ED.Text [0..1]	In the context of the LRA, in which a system is populating a QTY data value based on a locally held value, it is possible that the original text will not be known. This is an optional value in ISO 21090 and must not be relied upon to be populated in the LRA. In implementation there is expected to be specific rendering rules associated with individual user interface designs, for a given data point. It is a conforming systems responsibility to ensure that the actual value is semantically equivalent to the user input value.
uncertainty : QTY [0..1]	Expression of uncertainty expressed by this attribute is to some extent replicated by other datatypes in the specification (specifically URG). This uncertainty attribute describes uncertainty arising as a result of knowable limits to the precision of a measuring instrument - where there is, for example, a measured value and an expression of confidence that the value is correct. It is different from precision of the numeric representation in terms of significant digits. This is differentiated in ISO 21090 from an expression of uncertainty in the actual value of a measured attribute, which is represented as an uncertain range. In the LRA it is not felt that the expression of that confidence is significantly different. When required in medicine it is frequently expressed the same way, with expression of simple 95% (possibly asymmetric) confidence limits. This expression does not directly support such examples (e.g. relative risk = 0.2 with 95% confidence interval 0.01 - 0.21), is felt to potentially confuse, be unsupported by the majority of current clinical applications, and as such this attribute is deprecated from the LRA for the time being.
uncertaintyType : UncertaintyType [0..1]	As above this is deprecated in the LRA.

Constraints

Constraint Type	Name	Details
Invariant	no expression calculation in LRA	inv: expression.ocllsUndefined and expressionLanguage.ocllsUndefined
LRA Invariant	no data type uncertainty in LRA	inv: uncertainty.ocllsUndefined and uncertaintyType.ocllsUndefined
Invariant	uncertainty	inv: uncertainty.isNotNull implies (uncertainty.expression.isNull and

Constraint Type	Name	Details
		uncertainty.uncertainty.isNull and uncertainty.originalText.isNull) --- nullFlavor related
Invariant	No History or Update Mode	inv: noUpdateOrHistory(uncertainty) and noUpdateOrHistory(originalText) and noUpdateOrHistory(expression)

8.1.2 Class INT (Integer quantity)

Specialises: QTY

Realises:

This is implemented as described in ISO 21090.

Attributes

Attribute	Description
value : Integer [0..1]	This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	null or value	inv: isNull xor value.ocllsDefined
Invariant	no uncertainty	inv: uncertainty.isNull

8.1.3 Class INT.NonNeg (Integer quantity – not negative value)

Specialises: INT

Realises:

This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	not negative	inv: isNotNull implies value >= 0

8.1.4 Class INT.Pos (Integer quantity – positive value)

Specialises: INT.NonNeg

Realises:

This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	not negative	inv: isNotNull implies value > 0

8.1.5 Class CO (Coded ordinal quantity)

Specialises: QTY

Realises:

Coded ordinals in ISO 21090 represent scale results which have an implied order. This is a difficult subject on which there is much debate as the nature of coded ordinals is variable. Interpretation of an assessment scale result as a continuum does depend both on the precise nature of the scale and on the circumstances and the use case for which that interpretation is being used. It is valid, for example, to state that 2 assessments of GCS of the same patient 5 minutes apart of 13 and 12 are comparable numerically and there is clinical significance in performing this comparison. It is not the case, however, to say that two separate patients with a simultaneous GCS of 13 are clinically comparable.

Thus in the LRA coded ordinals should be only used to meet specific analysis use cases. If this is not the case then an enumerated set of ST representing all possible results is preferred, as this prevents the introduction of computable semantics that are not apparent to the user.

In the LRA CO MUST NOT be used as a parameter of IVL, as it is meaningless. They can however be used as a parameter for URG. The 21090 CO is represented unchanged in the LRA but its scope of use is limited.

This is implemented as described in ISO 21090.

Attributes

Attribute	Description
code : CD [0..1]	This is implemented as described in ISO 21090.
value : Integer [0..1]	This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	No History or Update Mode	inv: noUpdateOrHistory(code) -- nullFlavor invariants
Invariant	no code if null	inv: isNull implies code.octIsUndefined
Invariant	no value if null	inv: isNull implies value.octIsUndefined

Constraint Type	Name	Details
Invariant	must have a code or a value	inv: isNotNull implies (code.isNotNull or value.ocllsDefined)

8.1.6 Class REAL (Real quantity)

Specialises: QTY

Realises:

In ISO 21090 this value is used to represent real unit-less numbers, that are not expressions of ratio (see RTO). It is not used to represent physical quantities with units, for which subtypes of the PQV data type (itself a specialisation of QTY) is used. This is unchanged in the LRA

Attributes

Attribute	Description
precision : Integer [0..1]	This is implemented as described in ISO 21090.
value : Real [0..1]	This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	null or value	inv: isNull xor value.ocllsDefined

8.1.7 Class RTO (Ratio quantity)

Specialises: QTY

Realises:

This is as is in ISO 21090 other than the specific constraint to prevent the use of CO as a data type for the numerator and denominator.

Attributes

Attribute	Description
denominator : QTY [0..1]	Must not be coded ordinal (CO)
numerator : QTY [0..1]	Must not be coded ordinal (CO)

Constraints

Constraint Type	Name	Details
LRA Invariant	no coded ordinal ratios in LRA	inv: (not numerator.ocllsTypeOf(CO)) and (not

Constraint Type	Name	Details
		denominator.ocllsTypeOf(CO))
Invariant	numerator and denominator required	inv: isNull xor (numerator.isNotNull and denominator.isNotNull)
Invariant	no updateMode or History on RTO Attributes	inv: noUpdateOrHistory(numerator) and noUpdateOrHistory(denominator)
Invariant	no uncertainty	inv: uncertainty.isNull

8.1.8 Class MO (Monetary quantity)

Specialises: QTY

Realises:

This is implemented as described in ISO 21090.

Attributes

Attribute	Description
currency : Code [0..1]	This is implemented as described in ISO 21090.
precision : Integer [0..1]	This is implemented as described in ISO 21090.
value : Real [0..1]	This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	null or value	inv: isNull xor value.ocllsDefined
Invariant	null or currency	inv: isNull xor currency.ocllsDefined

8.1.9 Class PQV (Physical quantity value)

Specialises: QTY

Realises:

The class PQV is a quantity class which provides an ability to maintain a longitudinal transformation history through the source attribute.

Attributes

Attribute	Description
codingRationale : SET<CodingRationale>	The codingRationale set is optional in ISO 21090 in that it may be empty. It appears to refer to a translation attribute in the PQR and PQ subtypes which are deprecated in the LRA. CodingRationale's use is therefore also deprecated in the LRA in the

Attribute	Description
[1..1]	context of PQV
precision : Integer [1..1]	This is implemented as described in ISO 21090.
source : PQV [1..1]	This is implemented as described in ISO 21090.
value : Real [1..1]	This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
LRA Invariant	no codingRationale in LRA	inv: codingRationale->isEmpty

8.1.10 Class PQ (Physical quantity with UCUM units)

Specialises: PQV

Realises:

In ISO 21090 this represents an real value associated with a UCUM unit as a text string.

Attributes

Attribute	Description
translation : SET<PQR> [1..1]	in ISO 21090 the translation attribute specifies a set of equivalent measurements into non UCUM units (but not between UCUM units). This “translation” is in fact a prospective transformation. It is possible, as with the generalisation PQV to maintain a longitudinal record of performed transformations using the source attribute. A prospective translation in PQ has a single known use case, which is the dual expression of a drug dose in UCUM and DM&D units, which is described in the MiM HL7 data types implementation. For that purpose it is retained. It should not be used for any other purpose unless a clear implementation guide can be produced. This is implemented as described in ISO 21090.
unit : Code [0..1]	This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	null or value	inv: isNull xor value.ocIsDefined -- nullFlavor invariants
Invariant	no translation if null	inv: isNull implies translation->isEmpty --context PQ::isDifference(other : QTY): Boolean -- post: other.ocIsKindOf(PQ) and canonical.units.equals(other.ocAsTypeOf(PQ).canonical.units)

Constraint Type	Name	Details

8.1.11 Class PQ.Time (Physical quantity – time related)

Specialises: PQ

Realises:

This is unchanged in the LRA. It should only be used to represent a duration of time, which has neither end fixed, or for which the duration is totally independent of the timing of the event itself. Where one or other end of the duration can be sensibly placed at a point in time (even with great inaccuracy) an IVL<TS> data type should be used instead. PQ.Time should be used where length of time is being used as a measure of a class of event or a theoretical event. Examples could include the specification of duration of uterine contractions during labour, or an expected waiting time, or the result of a timed experiment.

This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	must be a unit of time	inv: canonical.unit = "s"

8.1.12 Class PQR (Physical quantity with coded units)

Specialises: PQV

Realises:

This class represents physical quantities that cannot be represented in UCUM unit, but are expressed using a concept in a standard terminology. The codingRationale attribute appears to be defined twice in the draft text of ISO 21090, but is not present as part of this class in the LRA, nor do we believe it is part of ISO 21090, this looks like a documentation error in 21090.

Attributes

Attribute	Description
unit : CD.CV [1..1]	This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
LRA Invariant	no codingRationale in LRA	inv: codingRationale.ocllsUndefined
Invariant	no originalText	inv: originalText.ocllsUndefined

Constraint Type	Name	Details
Invariant	null or value	inv: isNull xor value.ocllsDefined
Invariant	no source	inv: source.isNull
Invariant	no updateMode or History on PQR	inv: noUpdateOrHistory

8.1.13

class TimingRepresentation

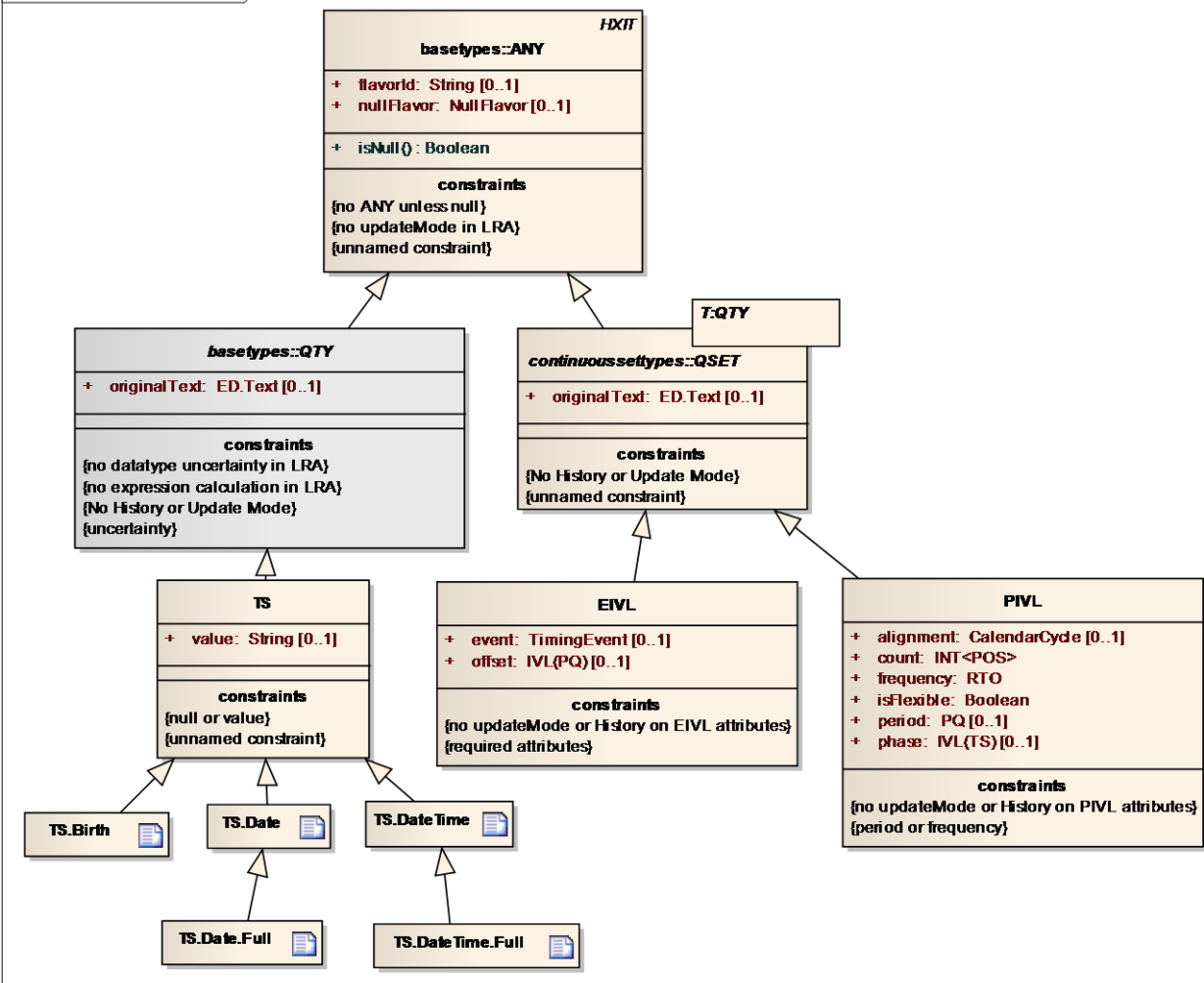


Figure 8: Timing Representation

8.1.14 Class TS (Time stamp)

Specialises: QTY

Realises:

TS is a string representation using standard notation of a point in time. This is implemented in the LRA as specified in ISO 21090. As with QTY both uncertainty

and uncertaintyType within this data type are difficult to distinguish from uncertainty represented by URG and UVL., and from an LRA implementation perspective where they are required, it is believed they should be managed as specifically stated uncertain dates or ranges. These constraints on uncertainty are present in the LRA profile of QTY and not formally restated here. Otherwise this class and its constraints are used as represented in ISO 21090.

This is implemented as described in ISO 21090.

Attributes

Attribute	Description
value : String [0..1]	This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Definitional		def: hasTimezone : Boolean = value.pos("+") > 0 or value.pos("-") > 0
Invariant	null or value	inv: isNull xor value.ocllsDefined

8.1.15 Class TS.Birth (Time stamp – birth date)

Specialises: TS

Realises:

This is implemented as described in ISO 21090.

8.1.16 Class TS.Date (Time stamp – date)

Specialises: TS

Realises:

This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	Date	inv: not hasTimezone and value.size <= 8

8.1.17 Class TS.Date.Full (Time stamp – fully specified date)

Specialises: TS.Date

Realises:

This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	Full Date	inv: value.size = 8

8.1.18 Class TS.DateTime (Time stamp – date and time)

Specialises: TS

Realises:

This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	DateTime	inv: (value.size <= 14) or (hasTimezone and value.size <= 19)

8.1.19 Class TS.DateTime.Full (Time stamp – fully specific date and time)

Specialises: TS.DateTime

Realises:

This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	Full DateTime	inv: value.size = 19 and hasTimezone

9 Continuous Set Data types

These data types provide support for collections of data.

9.1 Package Ira.datatypes.ISO 21090

9.1.1 Abstract Class QSET (Continuous set)

Specialises: ANY

Realises:

The QSET class in ISO 21090 defines continuous sets of information. It is made up of 4 operation or collection classes (QSI, QSD, QSU, and QSP) that allow for complex set membership expressions (intersection, difference, union and periodic hull (i.e. smallest containing continuous set), and 3 component subtypes (QSC, QSS, IVL and their subtypes PIVL and EIVL) QSET and its subtypes are

parameterised in ISO 21090 and they can be referred to by specifying the specific QTY type applying to them, which can be any QTY subtype in ISO 21090. In the LRA this is constrained to specifically exclude coded ordinals because of their uncertain nature. Therefore a QSET involving the data type REAL can be implemented within the LRA as QSET<REAL> or within an XML representation as QSET_REAL. The QSET involving time QSET<TS> is also known as GTS (or general timing specification). Within the LRA there is mainly a focus on two QSETs which will be examined specifically, these are QSC and IVL.

Attributes

Attribute	Description
originalText : ED.Text [0..1]	This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Definitional		inv: noUpdateOrHistory(originalText)
Invariant	No History or Update Mode	--- nullFlavor related

9.1.2 Class IVL (interval)

Specialises: QSET

Realises:

This class is used in ISO 21090 to specify a range of consecutive values of any ordered base data type (this is therefore parameterised to quantities - QTY). Its implementations such as IVL<TS> should be used for specifying a continuous range between two definable, realisable ends. It is not intended to be used to represent a single unknown value within an interval (for which use URG "" uncertain range). IVL should only be used to represent the existence of a continuous state between two delimiters. An example of IVL<TS> would include the timing of a medication infusion, or an admission period.

There are possible grey areas in the implementation of IVL compared with URG - an example being "take 2-4 tablets" could be modelled as an IVL<PQV> but the statement "he took between 12 and 24 tablets" clearly should only be modelled as URG<PQV>. The guidance for the LRA is that if a statement could theoretically be decomposed to a set of unitary statements separated by "or" the choice should be an URG, whereas if it can be broken down as "and" it should be an IVL. Thus both tablet related examples here should be modelled with URG (I.e. take 2 tablets OR take 3 tablets OR take 4 tablets).

The use of IVL with coded ordinals is explicitly prohibited.

Attributes

Attribute	Description
-----------	-------------

Attribute	Description
any : T [1..1]	<p>In ISO 21090 it is possible to also specify an IVL as a range that contains a specific point. So the IVL<INT> with attribute any=10 and width=5 represents all the intervals 5-10, 6-11, 7-12, 8-13, 9-14 and 10-15. This appears represent an specific type of uncertain interval which is different from uncertain range (URG). Within the LRA, as in ISO 21090, the use of IVL with only IVL.any and IVL.width attributes defined is allowable, and should be used with caution for the specification of loosely constrained time intervals. It is not however appropriate to use it for loosely defined time points for which URG<TS> is an appropriate choice.</p> <p>This is implemented as described in ISO 21090.</p>
high : T [0..1]	This is implemented as described in ISO 21090.
highClosed : Boolean [0..1]	This is implemented as described in ISO 21090.
low : T [0..1]	This is implemented as described in ISO 21090.
lowClosed : Boolean [0..1]	This is implemented as described in ISO 21090.
width : QTY [0..1]	<p>IVL start and end may be unspecified in ISO 21090 to allow for ranges of known size (using IVL.width) but undefined low and high end points. It is not allowable in ISO 21090 or the LRA to specify IVL.width and IVL.low, and leave IVL.high as implicitly defined. The statement of width is optional in the presence of high and low, but high and low must be present together.</p> <p>This is implemented as described in ISO 21090.</p>

Constraints

Constraint Type	Name	Details
LRA Invariant	no coded ordinal IVL in LRA	inv: (not low.ocllsTypeOf(CO)) and (not high.ocllsTypeOf(CO)) and (not any.ocllsTypeOf(CO)) and (not width.ocllsTypeOf(CO))
Invariant	co-occurrence rules	inv: (any_.isNotNull implies not (hasBounds or width.isNotNull)) and (any_.isNull implies (hasBounds xor width.isNotNull))
Definitional		def: hasBounds : Boolean = low.isNotNull or high.isNotNull
Invariant	no updateMode or History on IVL attributes	inv: noUpdateOrHistory(low) and noUpdateOrHistory(high) and noUpdateOrHistory(width)
Invariant	closed attributes only if limited	inv: (not low.isNotNull implies lowClosed.ocllsUndefined) and (not high.isNotNull implies highClosed.ocllsUndefined)
Invariant	null rules	inv: isNull xor (hasBounds or any_.isNotNull or width.isNotNull)

9.1.3 Class IVL.High (Interval – closed high)

Specialises: IVL

Realises:

This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	low	inv: low.ocllsUndefined and lowClosed.ocllsUndefined
Invariant	high	inv: isNotNull implies high.isNotNull and highClosed

9.1.4 Class IVL.Low (Interval – closed low)

Specialises: IVL

Realises:

This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	high	inv: high.ocllsUndefined and highClosed.ocllsUndefined
Invariant	low	inv: isNotNull implies low.isNotNull and lowClosed

9.1.5 Class IVL.Width (Interval - width only)

Specialises: IVL

Realises:

This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	low	inv: low.ocllsUndefined and lowClosed.ocllsUndefined
Invariant	high	inv: high.ocllsUndefined and highClosed.ocllsUndefined
Invariant	width	inv: isNotNull implies width.isNotNull

9.1.6 Class QSC (Coded quantity set)

Specialises: QSET

Realises:

This class is parameterised for quantities (QTY). It allows specification of a pre-existing QSET pattern using a code specification. The current coded content specified in ISO 21090 only specifies time patterns (as QSET<TS>) in a country dependent fashion. However QSC<TS> in the LRA may be used to specify prototypical time patterns such as medication regimens or timings of specific care plans.

Attributes

Attribute	Description
code : GTSabbreviation [1..1]	<p>The ISO 21090 constraint that QSC.code must be taken from the GTSAbbreviation HL7 code set is implemented in the LRA but using an LRA specific profile of the GTS vocabulary. Specific coded time patterns based on significant timings in the NHS may be developed for the LRA and referred to using that LRA specific vocabulary. Conforming systems may choose to represent the time pattern internally as they choose, but where a specific QSC<TS>.code is defined and mandated conforming systems MUST represent those times using an appropriate QSC<TS> to ensure computational equivalence between systems. Thus a QSC<TS> for “4 hour A&E waiting time” may be defined and where a conforming system has an equivalent timing it MUST represent that as a QSC<TS> on LRA conformant interfaces.</p> <p>This constraint is applied through the use of specific LRA vocabulary and is not formally expressed here.</p>

Constraints

Constraint Type	Name	Details
Invariant	not null	inv: isNotNull implies code.ocllsDefined

9.1.7 Class QSI (intersection of continuous sets)

Specialises: QSET

Realises:

This is implemented as described in ISO 21090.

Attributes

Attribute	Description
terms : Set(QSET(T)) [1..1]	This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	not null	inv: isNotNull implies terms->forAll(t t.isNotNull)
Invariant	size	inv: isNotNull implies terms->size >= 2

Constraint Type	Name	Details

9.1.8 Class QSD (Continuous set – difference)

Specialises: QSET

Realises:

This is implemented as described in ISO 21090.

Attributes

Attribute	Description
first : QSET(T) [0..1]	This is implemented as described in ISO 21090.
second : QSET(T) [0..1]	This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	not null	inv: isNotNull implies (first.isNotNull and second.isNotNull)

9.1.9 Class QSP (Continuous set – periodic hull)

Specialises: QSET

Realises:

This is implemented as described in ISO 21090.

Attributes

Attribute	Description
first : QSET(T) [0..1]	This is implemented as described in ISO 21090.
second : QSET(T) [0..1]	This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	not null	inv: isNotNull implies (first.isNotNull and second.isNotNull)

9.1.10 Class QSS (Set of continuous sets)

Specialises: QSET

Realises:

This is implemented as described in ISO 21090.

Attributes

Attribute	Description
terms : Set(T) [1..1]	This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	size	inv: isNotNull implies terms->size > 1
Invariant	not null	inv: isNotNull implies terms->forAll(t t.isNotNull)

9.1.11 Class QSU (Union of continuous sets)

Specialises: QSET

Realises:

This is implemented as described in ISO 21090.

Attributes

Attribute	Description
terms : Set(QSET(T)) [1..1]	This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	size	inv: isNotNull implies terms->size >= 2
Invariant	not null	inv: isNotNull implies terms->forAll(t t.isNotNull)

9.1.12 Class PIVL (Periodic interval)

Specialises: QSET

Realises:

Periodic intervals in ISO 21090 are the key way in implementing repeating events. PIVL is constrained to apply only to time, there is no other repeating dimension allowed. There is no periodic event equivalent and any periodic events are implemented as a periodic interval with zero duration for that interval using PIVL. The PIVL.phase.low.value attribute defines a known starting point of a repeating duration as a TS, the PIVL.phase.any.value defines the mid point of a range of possible starting points

Within the PIVL data type it is possible to under specify a periodic interval such that it has no set start date, for example, using under specification of the IVL data type. It does not appear possible to have an uncertain range of dates as the start point.

This is implemented as described in ISO 21090.

Attributes

Attribute	Description
alignment : CalendarCycle [0..1]	This is implemented as described in ISO 21090.
count : INT<POS> [1..1]	This is implemented as described in ISO 21090.
frequency : RTO [1..1]	This is implemented as described in ISO 21090.
isFlexible : Boolean [1..1]	This is implemented as described in ISO 21090.
period : PQ [0..1]	This is implemented as described in ISO 21090.
phase : IVL(TS) [0..1]	This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	period or frequency	inv: isNotNull implies (period.isNotNull xor frequency.isNotNull)
Invariant	no updateMode or History on PIVL attributes	inv: noUpdateOrHistory(phase) and noUpdateOrHistory(period)

9.1.13 Class EIVL (Event based interval)

Specialises: QSET

Realises:

This is implemented as described in ISO 21090.

Attributes

Attribute	Description
event : TimingEvent [0..1]	This is implemented as described in ISO 21090.
offset : IVL(PQ) [0..1]	This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	required attributes	inv: isNotNull implies (event.ocllsDefined)

Constraint Type	Name	Details
Invariant	no updateMode or History on EIVL attributes	inv: noUpdateOrHistory(offset)

9.1.14 Class GTS.BoundedPIVL (Bounded periodic time interval)

Specialises: QSET

Realises:

This is implemented as described in ISO 21090. It is variously described as GTS.BoundedPIVL or QSET<TS>.BoundedPIVL.

Constraints

Constraint Type	Name	Details
Invariant	GTS.BoundedPIVL 1	inv: terms->size = 2 -- def: ivl : IVL(TS) = terms->select(t t.ocIsKindOf(IVL(TS))) -- def: pivl : IVL(TS) = terms->select(t t.ocIsKindOf(PIVL(TS)))

10 Uncertainty Datatypes

These data types provide support for uncertain values. Note the support provided here, along with the support provided for uncertainty on QTY, provides support for quantitative uncertainty, not with the medical kinds of uncertainty encountered in clinical practice such as “likely to be x”, or a differential diagnoses.

10.1 Package Ira.datatypes.ISO 21090

10.1.1 Class URG (Uncertain range)

Specialises: QTY

Realises:

As mentioned in the context of ANY and IVL, URG is 21090s primary mechanism for dealing with values that are uncertain but known to fall within a specific range. These values are implicitly continuous in their nature and hence URG is parameterised by quantities (QTY). It is a general point that URG data types cannot be used in place of QTYS as part of, for example, IVL definitions.

Because of the multiple different expressions of uncertainty and embedded in the ISO 21090 data types, a decision was taken to deprecate uncertainty expressed within the data type, in favour of uncertain ranges or information model structures where specific requirements for representation of specific types of uncertainty are clear. This is in part a pragmatic conformance issue in that low level data type uncertainty is not known to be a feature of many current applications. This does

however mean that certain types of representation must be explicitly modelled in information structures, and hence cannot be part of the definition of other complex data types. Specifically the representation of a single value with a confidence range for measurement (e.g. systolic BP = 120mmHg +/- 15mmHg).

This is implemented as described in ISO 21090.

Attributes

Attribute	Description
high : T [1..1]	This is implemented as described in ISO 21090.
highClosed : Boolean [1..1]	This is implemented as described in ISO 21090.
low : T [1..1]	This is implemented as described in ISO 21090.
lowClosed : Boolean [1..1]	This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	nullFlavor	inv: nullFlavor = NullFlavor.UNK
Invariant	no inner attributes	inv: validTimeLow.ocllsUndefined and validTimeHigh.ocllsUndefined and controlActRoot.ocllsUndefined and controlActExtension.ocllsUndefined and flavorId.ocllsUndefined and nullFlavor.ocllsUndefined and updateMode.ocllsUndefined and expression.ocllsUndefined and originalText.ocllsUndefined and uncertainty.ocllsUndefined and uncertaintyType.ocllsUndefined

10.1.2 Class URG.High (uncertain range, bounded at high end)

Specialises: URG

Realises:

This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	low	inv: low.ocllsUndefined and lowClosed.ocllsUndefined
Invariant	high	inv: isNotNull implies high.isNotNull and highClosed

10.1.3 Class URG.Low (uncertain range, bounded at low end)

Specialises: URG

Realises:

This is implemented as described in ISO 21090.

Constraints

Constraint Type	Name	Details
Invariant	low	inv: isNotNull implies low.isNotNull and lowClosed
Invariant	high	inv: high.ocIsUndefined and highClosed.ocIsUndefined

11 References

HL7 Version 3 Standard: Data Types - Abstract Specification, Release 2, Normative Ballot 2 - May 2008 [<http://www.hl7.org>]

The openEHR Reference Model - Data Types Information Model Release 1.0.2 Revision 2.1.1. [<http://www.openehr.org/home.html>]

Measurement theory: Frequently asked questions [<ftp://ftp.sas.com/pub/neural/measurement.html>]

The Unified Code For Units Of Measure [<http://aurora.regenstrief.org/~ucum/ucum.html#para-29>]

Data elements and interchange formats — Information interchange — Representation of dates and times, INTERNATIONAL STANDARD ISO 8601, 2004-12-01.

Quantities and units —Part 11: Mathematical signs and symbols for use in the physical sciences and technology, INTERNATIONAL STANDARD ISO 31-11:1992

Information technology -- Open Systems Interconnection -- Remote Procedure Call (RPC), INTERNATIONAL STANDARD ISO 11578:1996

Information technology — General- Purpose Datatypes (GPD), INTERNATIONAL STANDARD ISO/IEC 11404:2007(E)

Health informatics – Harmonized data types for information interchange, INTERNATIONAL DRAFT STANDARD ISO 21090, 2007-09-24

Information technology -- Open Systems Interconnection -- Specification of Abstract Syntax Notation One (ASN.1), INTERNATIONAL DRAFT STANDARD ISO/IEC 8824:1990

W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes. [<http://www.w3.org/TR/xmlschema11-2/>]

UK Government Data Standards Catalogue (GDSC), Version 2.0, Agreed 01.01.02. GDSC [<http://www.govtalk.gov.uk/gdsc/html/default.htm>]

Codes for the representation of names of countries and their subdivisions - Part 1: Country codes, INTERNATIONAL STANDARD ISO 3166-1:2006

Royal Mail [<http://www.royalmail.com>]

NHS Data Model and Dictionary Version 3 [<http://www.datadictionary.nhs.uk>]

Information technology—Syntactic metalanguage—Extended BNF,
INTERNATIONAL STANDARD ISO/IEC 14977 : 1996(E)

New code system TelecommunicationCapabilities, Recommendation for HL7 RIM
Change For Harmonization During NOV2008, Recommendation Id INM-200811-04

Universal Unique Identifier, Open Group, CDE 1.1 Remote Procedure Call
specification, 1997 [<http://www.opengroup.org/onlinepubs/9629399/apdxa.htm>]

SNOMED CT® Transforming Expressions to Normal Forms Version 55 (31-Jan-07)
© 2002-2008 International Health Terminology Standards Development Organisation

SNOMED Clinical Terms Abstract Logical Models and Representational Forms
Version 6b (31-Jan-08) © 2002-2008 International Health Terminology Standards
Development Organisation

12 Appendices

A UML Primitive and Derived Types

UML Kernel Types

Name	Derivation	Description
Boolean	-	A binary <i>true</i> or <i>false</i> value type conforming to the definition of the <i>ISO 11404:2007</i> <code>boolean</code> data type.
Integer	-	Zero, a natural number (1, 2, 3, etc) or the negative of a natural number (-3, -2, -1, etc) conforming to the <i>ISO 11404:2007</i> <code>integer</code> data type.
Real	-	A rational (e.g. -1, -½, 0, 1%, 2, etc) or irrational (e.g. π, e, etc) number conforming to the definition of the <i>ISO 11404:2007</i> <code>real</code> data type.
String	-	Any sequence of characters conforming to the definition of the <i>ISO 11404:2007</i> <code>characterstring</code> data type.

UML Derived Types

Name	Derivation	Description
Code	String	A simple string that has a restricted set of possible values as defined in ISO 21090. Code is generally used where the code list is restricted by some external standard, such as an RFC.
Identifier	String	A sequence of non-whitespace characters symbolising the identity of an object or resource.
Expression	String	A controlled sequence of characters symbolising a data value or a relationship between data values.
Octet	Integer	Represents a number that can take values from 0-255 (also sometimes known as a byte) as defined in ISO 21090. This is defined to allow a formal UML definition of Binary..
Uri	String	A simple internet URL or URI as defined in ISO 21090. This is an extension of string but has a stereotype <<URI>> to assist with automated mappings to specific platforms, which may provide a specific type for dealing with binary content.

UML Extended Types

Name	Derivation	Description
XML	-	A placeholder for any XML content as defined in ISO 21090. This is used in the Data class to allow for either binary content or an XML object model.
Binary	-	A sequence of octets conforming to the definition of the <i>ISO 11404:2007</i> <code>octetstring</code> data type. This type is specifically created to allow UML declarations to refer to binary content (also sometimes known as streams) as ISO 21090.

B Representational forms of codes in the LRA

In ISO 21090 a clear policy decision has been made to move away from a structured representation of a code expression within the CD.code attribute of a data type as was the approach in HL7, to a representation where a code expression is held as plain string. The specification of structure and semantics of this string is firmly the responsibility of the defining terminology specification. The constraints specified within ISO 21090 are only that the string must be encoded as Unicode. Functional constraints however will also apply in that any representation of that string may be subject to manipulation such as character escaping in XML. Whilst these are implementation considerations, they are worth bearing in mind when determining how coded data is represented in the LRA.

The specification of the structure and semantics of the code string is therefore out of scope of ISO 21090, and various options are permissible as long as they are conformant to the underlying terminology specification. There are also examples within ISO 21090 which describe representations of different code systems. These are regarded as informative of an intention in a community of practice, and the LRA profile will follow them as closely as possible.

For this profile of ISO 21090 to be complete therefore there is a requirement to either

- profile the underlying specifications of representational forms of terminologies used in the LRA, where those are adequate, or
- to define a colloquial representational form, where they are not.

This, in turn, requires a defined scope of terminology usage within the LRA, which in turn a full set of functional requirements for the LRA based on a complete analysis of the information requirements of health and social care in the UK.

Given that pursuit of such a requirements-led methodology is ultimately unlikely to be successful, progress based on an iterative elaboration of core design principles has to be adopted. The result of this should be interpreted in the context of incompleteness. As such it is felt that decoupling the specification of data types from the representational forms of codes is desirable to allow evolution of one without impacting the implementation of the other. The following is therefore a first attempt to bound the problem in the context of the LRA and to define testable strategies for code representation in the early releases of the LRA, against the known LRA terminologies of SNOMED CT and dm+d. Subsequent releases of this document will devolve this appendix into an external resource.

Requirements

In the generic LRA profile of ISO 21090 codes are represented as strings and the primary vehicle for the human readable semantics is the mandatory CD.originalText attribute. There are some rigid constraints about what is, and is not, allowable for the use of a CD.code attribute instance employed in the LRA. These led to the following requirements for CD.code representations:

1. The code string must be able to represent both a single code and coded expression.

2. The status of the attestation of a CD instance is inherited from the attestation of the containing clinical statement. No assumptions can be made about the attested status of a code without reference to the containing clinical statement.
3. The code string must identify individual descriptive terms for each individual component of a coded expression, if and only if those descriptive terms were
 - a. attested by the user at the time of entry, or
 - b. used to render the text attested by the user, or
 - c. used to analyse the text attested by the user.
4. In the specific instance of automatically generated, unattested code expressions descriptive terms must not be supplied.
5. In the instance of codes undergoing mapping transformation it may be appropriate to represent a specific description in the target code system, depending on the nature of the map.
6. The code string must not be assumed to carry all the meaning of the concept independent of the code system it comes from. For example any term describing a disorder “Not otherwise specified” can only be defined in the context of what was specified in the rest of the terminology. It is not inconceivable that informational annotations of the code term may be later applied for clarification.
7. The code string must conform to a constraint of any defined specification normative for the code system.

Representation of SNOMED CT and dm+d codes and coded expressions in the LRA

Pre-coordinated codes (attested)

In SNOMED CT there are concepts and descriptions for those concepts which have human readable terms. As per requirement 1 above the code string must represent both concept and term. In the LRA pre-coordinated codes are regarded as a special case of a post coordinated expression, where there is no refinement. Within the draft specification “SNOMED CT Abstract Logical Model and Representational Forms” section 3.2.1 there is a specification for a SNOMED Compositional Grammar. The use of this representation of a SNOMED Concept is mandated for pre-coordinated code strings in the LRA, in accordance to requirement 7 above.

concept::= ws conceptId ws "|" ws term ws "|" ws

A concept is represented by a conceptId followed by a term enclosed by a pair of "|" characters. Whitespace before or after the conceptId is ignored as is any whitespace between the initial "|" characters and the first non-whitespace character in the term or between the last non-whitespace character and before second "|" character.

For the In the LRA profile of the SNOMED Compositional grammar required to meet requirement 3 implies that the “term” string defined as optional within the compositional grammar is made mandatory in this instance. It is noted that within the release data for SNOMED CT it is permissible for a term to contain a “|” character.

Within the LRA profile pipe characters in the term string must be replaced by “||” before inclusion in a code string. Code validation parsers will need to reverse this replacement.

Example:

```
232376006 | dislocated nasal septum |
414317003 | ||Gentamicin 240mg/100mL intravenous infusion |
```

It is not expected that an individual application will persist the code in the stated form internally, it may well make sense for an application to maintain an code expression library which maintains a mapping between an internal unique ID and the SNOMED CT coded expression as specified. Conformance to the LRA only requires that the interface expresses codes in this way.

Post-coordinated codes (attested – close to user)

The same LRA constraints apply for post-coordinated terms as for pre-coordinated, in that for attested CD instances the code string must specify the term, for every coded concept.

Example:

```
232376006 | dislocated nasal septum | :
    246112005 | severity | = 24484000 | severe |
    ,263502005 | clinical course | = 385315009 | sudden onset |
```

Normal form

The normal form representation of a SNOMED concept is expected to be generated from an original CD with no input from the user, as an intermediate step in determining subsumption or equivalence. It is asserted that in the LRA the normal form may be persisted for communication purposes and may be used during the communication process to determine some automatic behaviour, such as categorisation of the coded data. As such it may need to be persisted albeit temporarily, but it will not be directly attested. By its definition within the LRA the normal form will always have a source CD, which will define both the human readable view of the data, but also the clinical representational form.

The CD.code representation of the normal form is therefore subject to requirement 4 above and the representation of the string must not include reference to terms. And in this case the optional term string in the SNOMED CT Compositional Grammar is constrained away.

concept::= ws conceptId ws

A concept is represented by a conceptId. Whitespace before or after the conceptId is ignored.

The normal form can exist in several minor variations as described in the SNOMED CT® Transforming Expressions to Normal Forms document. Within the LRA the only

permitted form for the Normal form is the Canonical representation of the long normal form, which allows for a structurally and semantically specified form.

When a normal form is represented in a CD data type instance the originalText must be populated with the originalText of the source CD instance (consistent with the constraints of any generated CD). The codeSetVersion must specify the release of SNOMED CT that was used to generate the normal form, and not refer to the codeSet of the source CD instance.

The differentiation of the normal form as distinct from a directly entered and attested SNOMED CT expression is important for determining allowable possible behaviour. In the LRA therefore SNOMED CT expression normal form as will use an extension of the normal SNOMED OID as its codeSet specification (to be determined).

Example: (same expression as above expressed in canonical normal form)

```
243796009 : {246090004 = (232376006 : 246112005 = 24484000, 263502005 =  
385315009 {116676008 = 19130008, 363698007 = 68426009} ), 408729009 =  
410515003, 408731000 = 410512000, 408732007 = 410604004}
```

Representation of other code set codes

This section must be expanded to include details on other code systems within the scope of the LRA. Please comment on the code systems that need to be addressed in the scope of this section.